Języki Skryptowe

Dokumentacja projektu zaliczeniowego

Adam Rogalski
Wydział Matematyki Stosowanej
Informatyka 2020/2021 sem 3 st
Zadanie: Drapieżnik-Ofiara

Spis treści

1 Opis problemu do zrealizowania

Moim celem do zrealizowania było zaprojektowanie symulacji pościgu drapieżnika swojej ofiary. Ucieka ona po prostej, natomiast drapieżnik goni go również po prostej, lecz zmienia kierunek swojego pościgu po pewnym kroku czasu delta t, co czyni drogę pościgu drapieżnika krzywą. Użytkownik podaje koordynaty początkowe drapieżnika oraz ofiary, ich prędkości, krok czasu po którym, drapieżnik staje w miejscu aby skorygować drogę, czas po którym pościg skończy się na wskutek zmęczenia drapieżnika oraz współrzędne prostej po której ma poruszać się ofiara.

2 Model matematyczny zadania

2.1 Opis modelu matematycznego

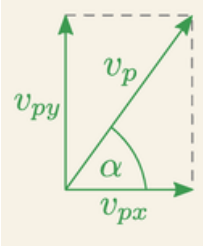
Aby stworzyć wzór prostej $y = ax+b$ potrzebujemy dwa punkty na płaszczyźnie lub współczynniki a oraz b prostej. W naszym przypadku obliczamy współczynniki prostej ofiary i drapieżnika ze wzorów:

$$a = \frac{y_d - y_o}{x_d - x_o}$$

$$b = y_d - (a * x_d)$$

gdzie: x_d, y_d-to współrzędne drapieżnika,
oraz x_o, y_o- to współrzędne ofiary

W ten sposób będziemy liczyć jeden raz prostą ofiary, oraz co każdy skok czasu prostą drapieżnika(przyjmując oczywiście nowy punkt położenia ofiary).
Kolejnym aspektem zadania jest prędkość naszych obiektów. Postanowiłem że zaimplementuję rozkład prędkości na wektory, aby nadać obiektom całkiem rzeczywistą prędkość w przypadku przestrzeni dwuwymiarowej.



$cos\alpha = \frac{V_{px}}{V_{py}}$
$V_{px} = V_p cos\alpha$
Kąt alfa jest nam znany dzięki zależności: $tg\alpha = a$,
więc: $\alpha = arctan(a)$
Teraz już tylko liczymy $cos\alpha$ i możemy policzyć składową Vpx, którą następnie dodamy do współrzędnej jednego z naszych obiektów i obliczymy drugą

współrzędną za pomocą wzoru $y = ax + b$
Co każdy krok czasu jest również liczony dystans jaki dzieli nasze dwa punkty, ze wzoru na odległość dwóch punktów na płaszczyźnie kartezjańskiej:
$d = \sqrt{(x_o - x_d)^2 + (y_o - y_d)^2}$

2.2 Przykład obliczeniowy

Przykład obliczeniowy dla danych:
$d = (10, -15), o = (-20, -40), V_d = 5, V_o = 3, \Delta t = 1, czas = 20,$
$Punkt2 = (-10, 10)$
Obliczamy współczynnik a oraz b prostej ofiary:
$a = \frac{-40-10}{-40-(-10)} = 5$
$b = -40 * \frac{-5}{-3} * (-20) = 60$
$y = 5x + 60$ $kat = arctan(a) \approx 1.373$
$cos\alpha = \approx 0.196$
$V_x = cos\alpha * V_o = \approx 0.58$
$y_o = a * x_o * b = \approx -42.94$
$dystans = \sqrt{(-20-10)^2 + (-40-(-15))^2} = \approx 36.43$ Działania będą takie same przy wyznaczaniu prostej dla drapieżnika, oczywiście ze zmieniającymi się współrzędnymi co skok czasu.

3 Algorytm

3.1 Słowny opis algorytmu

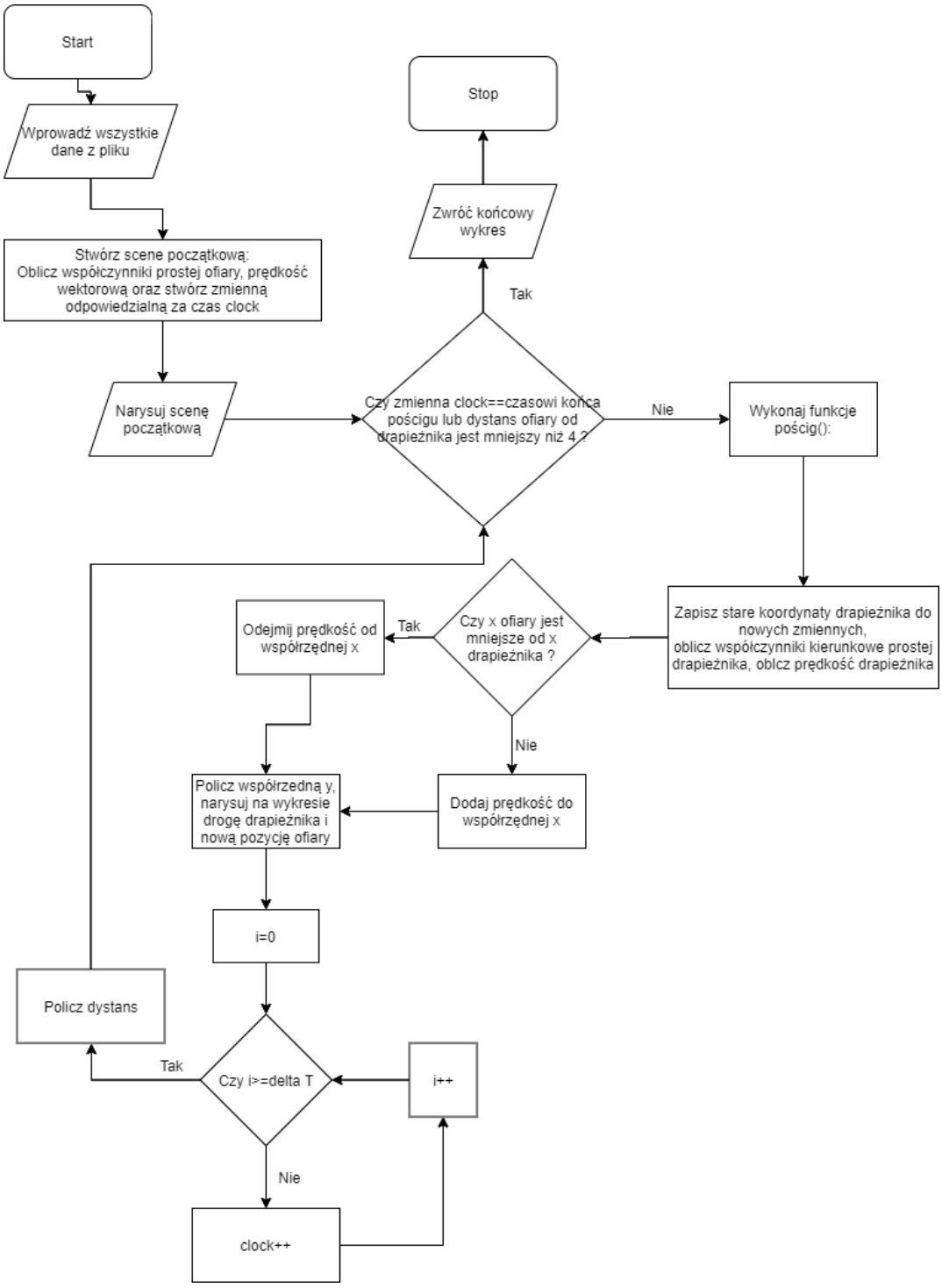
Algorithm 1: Pseudokod głównego algorytmu

```
Wczytaj wszystkie potrzebne dane z pliku
Stwórz i oblicz potrzebne zmienne, współczynniki prostej ofiary,
 prędkosć wektorową itd. Narysuj scenę początkową
while pościg się nie skończył do
    if pościg dobiegl końca then
        ustaw zmienną czykoniec=True
        break
    end
    wywwołaj funkcje pościg
end
Result: Wykres pościgu
```

Algorithm 2: Pseudokod algorytmu pościgu

Przechowaj stare koordynaty drapieżnika w nowych tymczasowych zmiennych
Oblicz współrzędne prostej drapieżnika
Oblicz wektor prędkości drapieżnika
if *koordynat x ofiary* $<$ *koordynatu x drapieżnika* **then**
    Od koordynatów drapieżnika i ofiary odejmij wartość prędkości wektorowej
else
    Do koordynatów drapieżnika i ofiary dodaj wartość prędkości wektorowej
end
Oblicz nowe koordynaty y ofiary i drapieżnika za pomocą wzoru prostej
Nanieś zmieny na wykres
for $i \leftarrow 0$ **to** ΔT 1 **do**
    zwiększ zmienna clock o 1
end
Oblicz dystans między punktami
Result: Wykres pościgu ale w pojedynczej jednostce czasu

3.2 Schemat blokowy



4 Implementacja

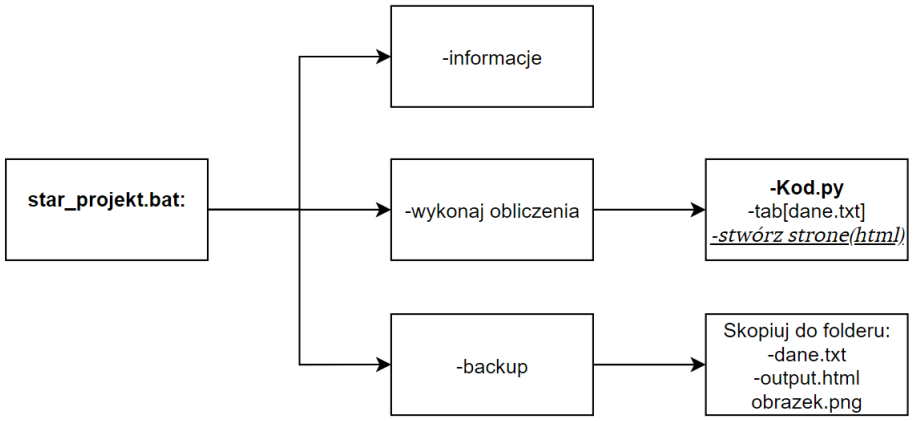
4.1 Budowa projektu

Start projekt.bat-skrypt powłoki systemowej sterujący całością, wyświetlający treść zadania, odpowiedzialny za tworzenie backupu.

kod.py- główny skrypt w którym znajduje się kod odpowiedzialny za wykonanie obliczeń.

Strona.py- skrypt odpowiedzialny za przedstawienie danych zadania w formie pliku html(strony internetowej).

Przepływ danych w projektcie:



4.2 Kod Źródłowy skryptu .bat

```
@echo off
setlocal enabledelayedexpansion

:main
set inputFile=dane.txt
set outputFile=wyjscie.html
cls
call :tytul
call :opcje
set /p input=wybierz opcje:
if !input!==1 call :info
if !input!==2 call :compute
```

```
if !input!==3 call :backup
if !input!==4 call exit


:tytul
echo Adam Rogalski - projekt "Drapieznik-Ofiara"
exit /b

:opcje
echo     1. Informacje ogolne
echo     2. Wykonaj program
echo     3. Wykonaj kopie danych
echo     4. Wyjdz
exit /b

:info
cls
echo Program rozwiazuje nastepujacy problem:
echo     Na plaszczyznie, w punkcie Pd znajduje sie
    drapieznik, ktory w chwili t0 (mozemy
echo przyjac, że    0    = 0) dostrzegl ofiare znajdującą
     się w punkcie P0 i w tej samej chwili rozpoczal
echo jej poscig z predkoscia vd. Ofiara, rowniez w tej
    samej chwili, rozpoczyna ucieczke po
echo pewnej prostej l0 z prędkością v0. Co pewien staly
    krok czasu delta t, drapieznik spoglada,
echo w ktorym miejscu znajduje sie ofiara i koryguje swa
    trase poscigu (drapieżnik rowniez
echo biegnie wzdluz prostej, jednak po kazdej korekcie,
    prosta ta może ulec zmianie).
echo Napisz program, ktory po zadaniu wspolrzednych
    punktow Pd i P0, predkosci Vd i V0,
echo parametrow prostej l0 oraz delta t, zwroci krzywe,
    po ktorych biegna drapieznik i ofiara (trasa
echo ucieczki ofiary jest prosta, a trasa pogoni
    lamana). Warunkiem zakonczenia poscigu jest
echo spelnienie jednego z warunkow: dogonienie ofiary lub
     zmeczenie sie drapieznika, co dzieje
echo sie po uplywie pewnego zadanego czasu t (kolejny
    argument programu).
echo W drugiej czesci rozbuduj program w taki sposob, aby
     ofiara, również co delta t,
echo korygowala trase ucieczki, tak aby biec w kierunku
    najkorzystniejszym dla uciekającej ofiary
echo (wowczas obydwie trasy beda lamanymi i nie trzeba
    zadawac parametrow prostej l0).
```

```
echo .
echo Nacisnij dowolny przycisk aby wrocic do menu
pause > NUL
goto :main
:compute
if not exist !inputFile! echo plik z danymi wejsciowymi
    nie istnieje, nacisnij przycisk aby przejsc dalej &
    pause > NUL & goto :main
python kod.py
echo Wykonano obliczenia
echo nacisnij dowolny przycisk aby przejsc dalej
pause > NUL
goto :main
:backup
set name=backup
if exist "C:\Users\Adam\anaconda3\envs\pythonProject\
    backup" echo Backup już istnieje, przenies lub usun go
     aby kontynuowac & echo Wcisnij dowolny przycisk, ay
    wrocic do menu & pause > NUL & goto main
mkdir "C:\Users\Adam\anaconda3\envs\pythonProject\backup"
if exist !inputFile! copy !inputFile! !name!
if exist !outputFile! copy !outputFile! !name!
if exist wynik.png copy wynik.png !name!
echo.
echo Kopia zapasowa znajduje sie w folderze !name!
echo Wcisnij dowolny przycisk, aby wrocic do menu
pause > NUL
goto main
:exit
exit 0
```

[language=iPython]

4.3 Kod Źródłowy skryptu obliczeniowego .py

```
import  math
import Strona
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

class Projekt:

    def __init__(self,x_drap,y_drap,x_of,y_of,Vd,Vo,del_t
        ,czas_t,x_dest,y_dest):
        #incjalizacja wszystkich danych
        self.x_drap=x_drap
```

```
        self.y_drap=y_drap
        self.x_of=x_of
        self.y_of=y_of
        self.Vd=Vd
        self.Vo=Vo
        self.del_t=del_t
        self.czas_t=czas_t
        self.x_dest=x_dest
        self.y_dest=y_dest
        #współczynniki a i b prostej po której biegnie
            ofiara
        self.a=(y_of-y_dest)/(x_of-x_dest)
        self.b=y_of-((y_of-y_dest)/(x_of-x_dest))*x_of
        kat = np.arctan(self.a) #kat potrzebny do
            wyznaczenia wektora predkosci ofiary (tan alfa
            =a)
        cos = math.cos(kat)  #cos tegoż kąta
        self.Vxo = cos * self.Vo #wektor prędkości ofiary
        self.czykoniec=False    #wartość boolowska
            odpowiedzialna na sterowaniem końca pościgu
        self.clock=0    #licznik czasu, zauważyłem że
            najprościej czas będzie liczyć jako ilość "
            skoków" wykonanych przez drapieżnika
        self.dist=math.sqrt((self.x_of-self.x_drap)**2+(
            self.y_of-self.y_drap)**2)


#poczatkowa scena i elementy urozmaicające nasz wykres
    def scenapoczatkowa(self):
        plt.plot([self.x_drap], [self.y_drap], 'go')
        plt.plot([self.x_of], [self.y_of], 'yo')
        plt.axline([self.x_of, self.y_of], [self.x_dest,
            self.y_dest], color='red')

        plt.annotate('START', xy=(self.x_drap, self.
            y_drap), xytext=(self.x_drap+1, self.y_drap+1)
            ,
                     arrowprops=dict(facecolor='green'),
                     )
        plt.annotate('START', xy=(self.x_of, self.y_of),
            xytext=(self.x_of -2, self.y_of - 2),
                     arrowprops=dict(facecolor='yellow'),
                     )
        plt.xlabel("Oś X")
        plt.ylabel("Oś Y")
        plt.title("Wykres ucieczki ofiary przed drapież
            nikiem")
        plt.grid(True)
```

```
        greenpatch=mpatches.Patch(color="green",label="
            Trasa Drapieżnika")
        redpatch = mpatches.Patch(color="red", label="
            Trasa Ofiary")
        plt.legend(handles=[redpatch,greenpatch])

    def poscig(self):

        #stare współrzędne drapieżnika muszą zostać
            zapisane do zmiennych
        #aby można było później narysować wykres
        x = self.x_drap
        y = self.y_drap

        #współczynniki prostej drapieżnika
        a_drap = (self.y_drap - self.y_of) / (self.x_drap
            - self.x_of)
        b_drap = self.y_drap - (a_drap * self.x_drap)

        #obliczanie wektora prędkości dla drapieżnika
        kat=np.arctan(a_drap)
        cos=math.cos(kat)
        Vdx=cos*self.Vd

        #te warunki sprawiają, że bestia zachowuje się
            trochę inteligętniej,
        #jeśli współrzędne x ofiary są mniejsze od drapie
            żnika to dodaje się do wspórzędniej
        #i na odwrót, tak aby uciekinier oddalał się od
            ofiary a drapieżnik przybliżał do niej.
        if self.x_of<self.x_drap:
            self.x_drap=self.x_drap-Vdx
            self.x_of = self.x_of - self.Vxo
        else:
            self.x_drap=self.x_drap+Vdx
            self.x_of = self.x_of + self.Vxo

        #obliczanie drugiej współrzędnej po dodaniu
            wektora prędkości do współrzędnej x
        self.y_drap=a_drap*self.x_drap+b_drap
        self.y_of=self.a*self.x_of+self.b
        plt.plot([x,self.x_drap],[y,self.y_drap],color='
            green')
        plt.plot([self.x_drap], [self.y_drap], 'go')
        plt.plot([self.x_of], [self.y_of], 'yo')
        circle = plt.Circle((self.x_drap, self.y_drap),
            4, color="pink",fill=False) #zasięg ataku
            drapieżnika
        plt.gca().add_patch(circle)
```

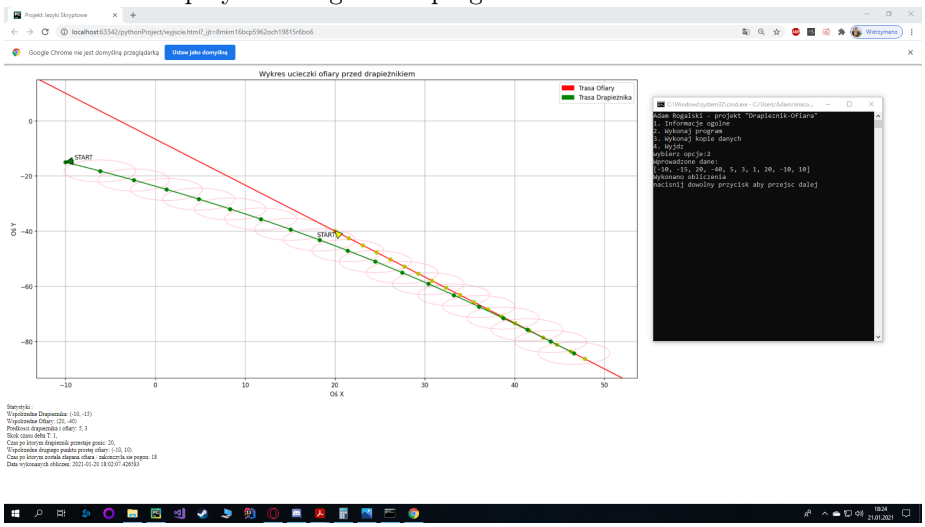
```
        # pętla która tworzy iluzje czasu(jeżeli ustawimy
            delta t na 2 i limit czasu 20 to drapieżnik
           wykona 10 ruchów bo 20/2=10
        for i in range(0, self.del_t):
            self.clock+=1

        # dystans drapieżnika od ofiary potrzebny do okre
           ślenia czy ofiara została już złapana
        self.dist = math.sqrt((self.x_of - self.x_drap)
           ** 2 + (self.y_of - self.y_drap) ** 2)


    def ifover(self):
        if self.clock==self.czas_t:
            self.czykoniec=True
        if self.dist<=4:
            self.czykoniec=True

    def getclock(self):
        return self.clock
try:
    Dane=open("dane.txt","r")
except IOError:
    print("Nie znaleziono pliku z danymi")
tab=[]
for lines in Dane.readlines():
    tab.append(int(lines))                      #
       wydobycie danych z pliku
Dane.close()
print("Wprowadzone dane:")
print(tab)

prjkt=Projekt(tab[0],tab[1],tab[2],tab[3],tab[4],tab[5],
   tab[6],tab[7],tab[8],tab[9])
prjkt.scenapoczatkowa()
#pętla sterująca końcem
while not prjkt.czykoniec:
    prjkt.ifover()
    if prjkt.czykoniec:
        break
    prjkt.poscig()


figure=plt.gcf()
figure.set_size_inches(18,9)
plt.savefig("wynik.png",bbox_inches='tight',dpi=120) #
   zapisanie obrazka
#tworzenie strony
Strona.stworzstrone(tab[0],tab[1],tab[2],tab[3],tab[4],
   tab[5],tab[6],tab[7],tab[8],tab[9],prjkt.getclock())
```

4.4 Kod Źródłowy skryptu wykonawczego

```
from datetime import datetime


def stworzstrone(x_drap, y_drap, x_of, y_of, Vd, Vo,
    del_t, czas_t, x_dest, y_dest, clock):
    f = open("wyjscie.html", "w")

    f.write("<!DOCTYPE html>\n")
    f.write("""<html lang="pl">\n""")
    f.write("<head>\n")
    f.write("""\t<meta charset="UTF-8">\n""")
    f.write("\t<title>Projekt Jezyki Skryptowe</title>\n"
        )
    f.write("</head>\n")
    f.write("<body>\n")
    f.write("\t<img src='wynik.png'>\n")
    f.write("<br/>Statystyki : <br/>\n")
    f.write("Wspolrzedne Drapieznika: (" + str(x_drap) +
        ", " + str(y_drap) + ")<br/>")
    f.write("Wspolrzedne Ofiary: (" + str(x_of) + ", " +
        str(y_of) + ")<br/>")
    f.write("Predkosci drapieznika i ofiary: " + str(Vd)
        + ", " + str(Vo) + "<br/>")
    f.write("Skok czasu delta T: " + str(del_t) + ", " +
        "<br/>")
    f.write("Czas po ktorym drapieznik przestaje gonic: "
         + str(czas_t) + ", " + "<br/>")
    f.write("Wspolrzedne drugiego punktu prostej ofiary:
        (" + str(x_dest) + ", " + str(y_dest) + ")<br/>")

    f.write("Czas po ktorym zostala zlapana ofiara /
        zakonczyla sie pogon: " + str(clock) + "<br/>")
    f.write("Data wykonanych obliczen: " + str(datetime.
        now()))
    f.write("</body>\n</html>")

    f.close()
```

4.5 Przykładowy test

Zrzut ekranu z przykładowego testu programu:



5 Podsumowanie

5.1 Co zostało zrobione

Wykonana została sieć skryptów która z poziomu powłoki systemu z pomocą pliku z danymi, pozwala obliczyć i zwrócić trasę pogoni drapieżnika za ofiarą w układzie współrzędnych i zwrócić dane w postaci wykresu oraz statystyk na pliku strony internetowej. Największym problemem w projekcie było zespolenie ze sobą wielu składowych, zmiennych programu tak aby tworzyły jedną, logiczną całość. Odniosłem wrażenie, że zadanie ma szerokie pole interpretacji i mogło zostać zrealizowane na dużo sposobów. Przetestowałem program na systemach Windows 10 oraz Windows 7.

5.2 Co można zaimplementować ?

Do zaimplementowania została druga część zadania, można by było również głębić dokumentację "matplotlib" i zaimplementować którąś z funkcji na którą sam nie wpadłem.