Symulacja wpływu myślistwa na liczebność dzików w miastach

Filip Piskorski, Ewelina Badeja, Piotr Kądziela

Kraków 2023

Spis treści

Wstęp

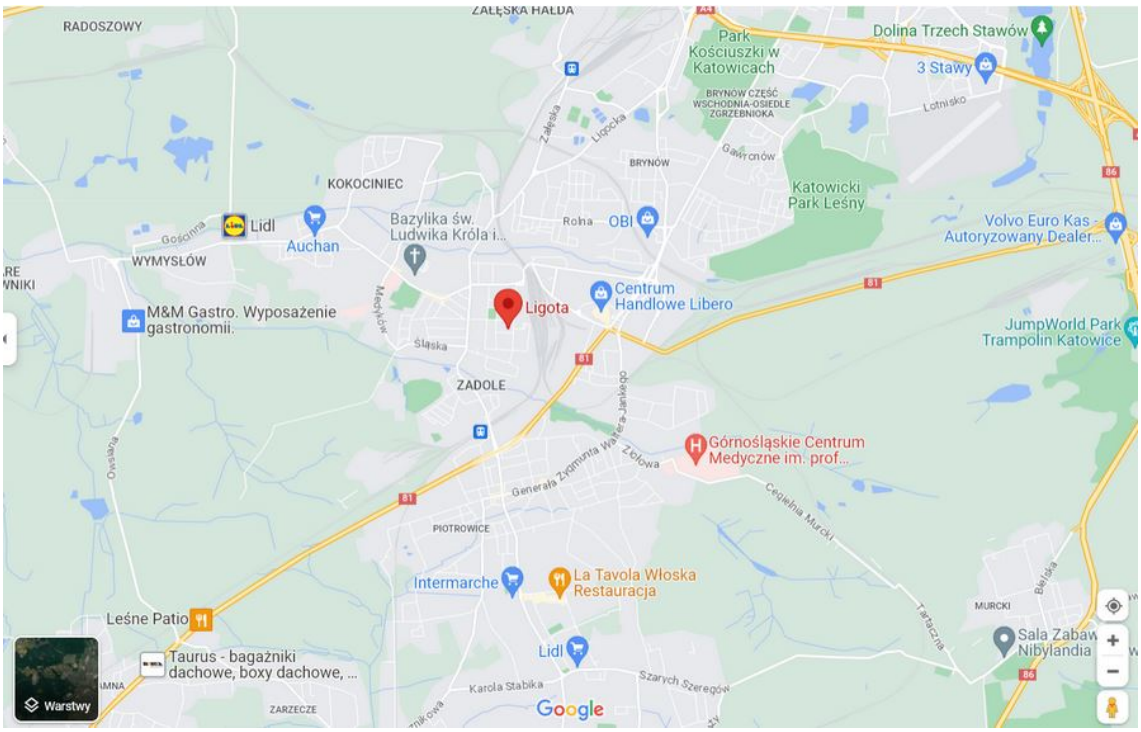
W wielu miejscowościach w Polsce problemem są dziki, które wabione zapachem jedzenia wchodzą na teren miast, stwarzając tym samym zagrożenie dla mieszkańców. Z pomocą programu, bazującego na modelu dyskretnym, zasymulowaliśmy wpływ obecności myśliwych na liczebność dzików na terenie Ligoty (tj. jednej z dzielnic Katowic).

Rozdział 1

Preliminaria

1.1 Opis problemu

Ligota to dzielnica Katowic, na terenie której znajdują się zarówno tereny miejskie, jak i leśne 1.1. Podobnie jak wiele innych miejscowości graniczących z lasem, występuje tam problem dzików, które wabione zapachem jedzenia wchodzą do miast (1.2,1.3). Jest to niebezpieczne dla mieszkańców Katowic, ponieważ poza szkodami, które są wyrządzane przez dziki w przestrzeni publicznej jak i prywatnych ogródkach, sprowokowane zwierzęta mogą atakować ludzi.



Rysunek 1.1: Mapa przedstawiająca badany obszar. Źródło: [2]

Dziki upodobały sobie miejskie osiedla

Dariusz Brombosz 2021-03-05 | 13:51 SKOMENTUJ (0)

Śląskie miasta coraz częściej mierzą się z „dzikim problemem". W katowickiej dzielnicy Ligota, dziki spacerują wokół osiedlowych alejek i wchodzą do śmietników.

Rysunek 1.2: Fragment artykułu o dzikach w miastach. Źródło: [1]

Strona główna / Kategorie / Katowice: Poród w ogródku na Ligocie! [ZDJĘCIA] Na świat przyszły ...malutkie dziki!

Kategorie

Katowice: Poród w ogródku na Ligocie! [ZDJĘCIA] Na świat przyszły ...malutkie dziki!

importer · 23 kwietnia 2015 0 1 minuta

Najnowsze Najczęściej czytane

9 maja 2023
Śląskie: Wójt gminy Jaworze wypadł z okna hotelu na Sardynii. Nie żyje

9 maja 2023
Jaworzno: Volkswagen golf a w nim kompletnie pijany nastolatek

9 maja 2023
Pogrzeb 8-letniego Kamilka w najbliższą sobotę

Rysunek 1.3: Fragment kolejnego artykułu o dzikach w miastach.Źródło: [4]

Jednym ze sposobów walki z opisanym problemem jest odstrzał dzików przez myśliwych. Sporo osób uważa ten sposób za dobre rozwiązanie, jednak ta metoda ma też swoich przeciwników. W histori myślistwa zdarzały się już wypadki, gdzie myśliwy postrzelił człowieka 1.4.

W ciężkim stanie do szpitala trafił myśliwy, którego z broni myśliwskiej postrzelił we wtorek na Podkarpaciu podczas polowania jego kolega. 35-latek, który pociągnął za spust, najprawdopodobniej pomylił znajomego z dzikiem.

Do zdarzenia doszło we wtorek w okolicach Zawadki Rymanowskiej na Podkarpaciu. Około godziny 14.30 jeden z myśliwych poinformował służby, że przypadkowo postrzelił swojego znajomego.

Rysunek 1.4: Wiadomości telewizjo TVN24 o człowieku, postzelonym przez myśliwego. Źródło: [3]

Nasz projekt ma na celu zbadanie wpływu myślistwa na liczebność dzików w miastach. Badany będzie czas, po którym pewna liczba ustawionych w dany sposób myśliwych zminimalizuje liczbę dzików na interesującym nas terenie do bezpiecznego poziomu. Następnie wynikowy czas będzie zestawiany z liczbą myśliwych, którzy byli potrzebni, oraz z tym, czy aspekty prawne pozwalają na dane ich rozłożenie.

1.2 Oczekiwania wobec symulacji

Program powinien w miarę możliwości jak najbardziej realistycznie symulować oddziaływanie na siebie nawzajem dzików, myśliwych oraz badanego terenu. Żeby spełnić te oczekiwania, każdy z wymienionych elementów symulacji powinien posiadać pewne funckjonalności:

- Dziki:
  - Zjadanie jedzenia

  - Ocena atrakcyjności okolicy (m. in. pod względem dostępności jedzenia)
  - Przemieszczanie się na bardziej atrakcyjne tereny

- Myśliwi:
  - Wabienie dzików
  - Strzelanie do dzików
  - Pudłowanie z pewnym prawdopodobieństwem
  - Bycie okresowo obecnym w ambonach strzelniczych

- Teren:
  - Podział na las, miasto i tereny podmokłe
  - Zwiększanie ilości jedzenia w lesie wraz z upływem czasu
  - Zwiększanie atrakcyjności miasta w określonych sytuacjach

1.3 Teza

Myślistwo może pomóc w walce z problemem dzików w mieście, jednak potrzeba by do tego bardzo dużo myśliwych i musieli by oni być rozstawieni niebezpiecznie blisko miasta.

Rozdział 2

Udoskonalanie symulacji na podstawie pozyskanych danych

2.1 Zachowania dzików

Ze strony kół łowieckich dowiedzieliśmy się, że ”Dziki żyją w ugrupowaniach zwanych watahami.”(Rys.2.1). Nasz program będzie więc musiał uwzględniać kilka dzików, poruszających się razem.

DZIK – zaliczany jest do zwierzyny ***grubej*** i ***czarnej***. Według wieku dzielimy je na ***warchlaki***, ***przelatki***, ***wycinki***, ***odyńce*** i ***lochy***. Młoda samica to ***loszka*** a stara doświadczona to ***samura***. Odyniec, który bierze udział w huczce to ***gamrat***. W obecnych czasach loszki rzucają mioty już jako ***przelatki*** czyli w 2 roku życia. Dziki żyją w ugrupowaniach zwanych ***watahami***. Warchlaki ze względu na umaszczenie sukni nazywa się ***pasiakami***. Aktywność dzików przypada na poranek, wieczór i noc. W ciągu dnia odpoczywają w ***barłogu***. Lubią ***tarzać się*** i ***babrać*** w ***brochowisku*** i stąd ich dodatkowe ich nazewnictwo – ***błociarz***, ***smoluch***, ***smolak***, ***czarnuch*** itp.

Rysunek 2.1: Fragment strony edukacyjnej kół łowieckich .Źródło: [5]

Na tej samej stronie znaleźliśmy również informację, że ”W ciągu dnia odpoczywają w barłogu. Lubią tarzać się i babrać w brochowisku”, tzn. przebywać na terenach podmokłych. Na badanym terenie znajduje się kilka jezior, wokół których rozciągają się tereny bagienne. Będziemy zatem musieli uwzględnić to, że dla dzików tereny te będą bardzo atrakcyjne.

Kolejną przydatną informacją, którą udało nam się wyszukać, jest zasięg węchu

dzika. Według Policji Warmińsko-MAzruskiej "Dzik ma bardzo wyczulony węch (w odległości 500 metrów może wyczuć człowieka)..."(Źródło: [6]), dlatego też każda wataha decydując gdzie pójść będzie badać atrakcyjność terenów odległych o maksymalnie 500m.

2.2 Zachowania myśliwych

Po dyskusji przeprowadzonej z grupą ćwiczeniową doszliśmy do wniosku, że pomimo dużego zasięgu niektórych broni palnych używanych przez myśliwych, realna odległość, z której można upolować dzika w lesie, to maksymalnie 200m (lub nawet mniej).

Dodatkowo uzgodniliśmy, że myśliwi nie będą się przemieszczać- każdy będzie miał przypisane jedno miejsce na mapie, w którym co jakić czas będzie obecny i będzie mógł wtedy z pewnym prawdopodobieństwem ustrzelić dzika, który znajdzie się w zasięgu strzału.

Każdy myśliwy pojawia się na polu wraz z pewną ilością jedzenia- ma to symbolizować przynętę. Kiedy myśliwy zabije dzika, pole, na którym znajdował się dzik, powinno mieć na pewien czas obniżoną atrakcyjność. Sprawi to, że reszta watahy nie będzie chciała przwbywać na tym polu i zasymuluje ucieczkę.

2.3 Model badanego terenu

Zazwyczaj hałaśliwe miasto nie jest najlepszym miejscem do życia dla dzików. Jeden z członków naszego zespołu, który mieszka niedaleko badanego terenu, zaobserwował jednak, że w dniu wywozu śmieci dziki wabione ich zapachem potrafią wyjść z lasu i spacerując po osiedlach wywracać śmietniki. Nasz model powinien przez to uwzględniać dni wywozu śmieci, podczas których atrakcyjność miasta będzie znacząco rosła.

Celem naszego projektu jest przeprowadzenie symulacji, opierając się o model dyskretny, dlatego badany fragment terenu zostanie przedstawiony jako macierz kwadratowa o pewnych wymiarach. Dane pole macierzy może mieć jeden z czterech typów: las, miasto, teren podmokły lub myśliwy. Dodatkowo na każdym z tych pól może znajdować się wataha dzików. W jednej iteracji dziki mogą się przemieścić o jedno pole w dowolnym kierunku. Interesujący nasz obszar ma kształt kwadratu o boku 6.4 km, zdacydowaliśmy więc, że jedno pole w macierzy będzie symbolizowało 1ha (czyli $10000m^2$).

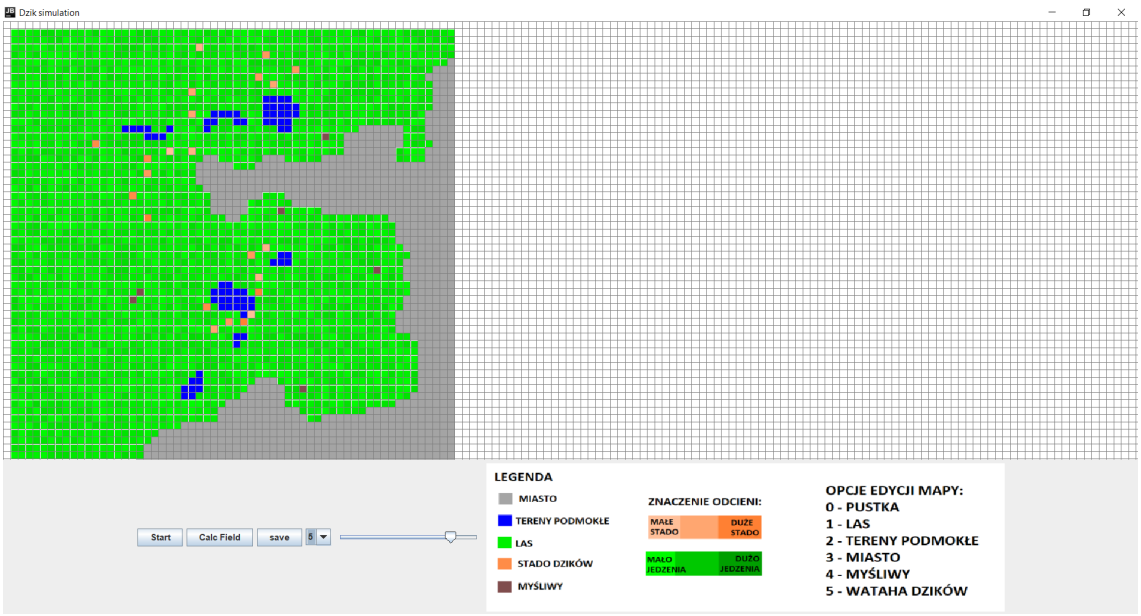
Rozdział 3

Dokumentacja kodu

Program (głównie GUI) był bazowany na kodzie dostarczonym przez prowadzących przedmiot.

3.1 Instrukcja korzystania z programu

Żeby rozpocząć symulację, użytkownik powinien wybrać z listy odpowiednio dziki lub myśliwych i umieścić ich na mapie 3.1. Jest możliwość edycji mapy, jednak mapa automatycznie wczytywana bardzo dobrze obrazuje badany teren, więc nie zaleca się tego robić.



Rysunek 3.1: Widok programu w trybie pełnoekranowym z perspektywy użytkownika.

Następnie można kliknąć przycisk ”Start”i obserwować zachowania dzików. Dane statystyczne są automatycznie zapisywane do pliku out.txt.

3.2 Opis poszczególnych klas

3.2.1 Program

Klasa pozwalająca na uruchomienie programu (Rys.3.2) .

```
public class Program extends JFrame {

    private static final long serialVersionUID = 1L;
    private GUI gof;

    public Program() {
        setTitle("Dzik simulation");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        gof = new GUI(this);
        gof.initialize(this.getContentPane());
        this.setSize(720, 720);
        this.setVisible(true);
    }

    public static void main(String[] args) { new Program(); }
}
```

Rysunek 3.2: Kod klasy Program.

3.2.2 Board

Kluczowa dla działania programu klasa, reprezentująca badany teren. Przechowuje ona listę dzików, mapę, stałe, regulujące m. in. atrakcyjność danych pól czy częstotliwość wywozu śmieci (Rys.3.3).

Najważniejszą funckją klasy Board jest funkcja Iterate (Rys.3.4). Jest ona odpowiedzialna za aktywowanie dnia wywozu śmieci, dawanie sygnału myśliwym do strzelania oraz dawanie sygnału dzikom do poruszania się i jedzenia.

Kolejnymi ważnymi metodami klasy Board są funkcje do zapisywania mapy (Rys.3.5) oraz funkcja do obliczania atrakcyjności pól (Rys.3.6).

```java
public class Board extends JComponent implements MouseInputListener, ComponentListener {
    private static final long serialVersionUID = 1L;
    private static final int HUNTER_SENSE_RADIUS = 5;
    private static final int STATS_SAVE_PERIOD = 20;
    public Point[][] points;
    Random rand = new Random();
    private int size = 10;
    public int editType = 0;
    private static int ctr = 0;
    public final Map<Point, Set<Dzik>> dziks = new HashMap<Point, Set<Dzik>>();
    private int averageAttractiveness = 0;
    private List<IterationStatistics> stats = new ArrayList<IterationStatistics>();

    public static final int MAX_SIZE = 60;

    private static final int SFMAX = 10000000;
    private static final int MIASTO_UNATTRACTIVENESS = 10000;
    private static final int BAJORA_ATTRACTIVENESS = 100000;
    private static final int LAS_ATTRACTIVENESS = 1000;
    private static final int OTHER_DZIKS_UNATTRACTIVENESS = 20000;
    public static final int DZIK_SENSE_RADIUS = 5;

    private static final int GARBAGE_COLLECTION_FREQUENCY = 168;
    //how often is garbage collected (in iterations)

    private static final int GARBAGE_COLLECTION_LENGTH = 24;
    //how long is garbage collected (in iterations); must be lower than GARBAGECOLLECTIONFREQUENCY

    private static final float HUNTER_KILL_PROPABILITY = 0.1f;
```

Rysunek 3.3: Zmienne oraz stałe, przechowywane przez klasę Board.

```
public void iteration() {

    stats.add(new IterationStatistics(ctr, countDziks(), countDziksInCities(), countDziksInTheWild(), countFood()));
    if(ctr%STATS_SAVE_PERIOD == 0){
        saveStats();
    }

    if(ctr % GARBAGE_COLLECTION_FREQUENCY == 0){
        for (Point[] row : points) {
            for (Point point : row) {
                if (point.type == 3)
                    point.triggerGarbageCollection();
            }
        }
    }
    if(ctr % GARBAGE_COLLECTION_FREQUENCY == GARBAGE_COLLECTION_LENGTH){
        for (Point[] row : points) {
            for (Point point : row) {
                if (point.type == 3)
                    point.eatAllFood();
            }
        }
    }

    theHunterKills();


   calcAverageAttractiveness();
   calculateStaticField();
   List<Dzik> dzikList = new ArrayList<Dzik>();
   for(Set<Dzik> dzikSet : dziks.values())
       dzikList.addAll(dzikSet);
   for(Dzik dzik : dzikList)
               dzik.move();
   for(Dzik dzik : dzikList)
               dzik.eat();

   for (int x = 0; x < points.length; ++x) {
       for (int y = 0; y < points[x].length; ++y) {
           points[x][y].growFood();
       }
   }

   this.repaint();
   ctr++;
}
```

Rysunek 3.4: Kod funkcji Iterate.

```
public void saveMap(){
    try {
        File file = new File("resources/map.txt");
        if(!file.exists()){
            file.createNewFile();
        }
        FileWriter fw = new FileWriter(file.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(MAX_SIZE + " " + MAX_SIZE + "\n");
        for (int y = 1; y <= MAX_SIZE; ++y) {
            for (int x = 1; x <= MAX_SIZE; ++x) {
                bw.write(points[x][y].type + " ");
            }
            bw.write("\n");
        }
        bw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Rysunek 3.5: Kod funkcji saveMap, odpowiedzialnej za zapisywanie w pliku zmian na mapie.

```
private void calcAverageAttractiveness(){
    int sum = 0;
    for (int i = 1; i < points.length-1; ++i) {
        for (int j = 1; j < points[i].length-1; ++j) {
            sum -= points[i][j].type == 3 ? MIASTO_UNATTRACTIVENESS : 0;
            sum += points[i][j].type == 2 ? BAJORA_ATTRACTIVENESS : 0;
            sum += points[i][j].type == 1 ? LAS_ATTRACTIVENESS : 0;
            sum -= allDziksHere(i,j) * OTHER_DZIKS_UNATTRACTIVENESS;
        }
    }
    averageAttractiveness = sum/ MAX_SIZE / MAX_SIZE;
}
```

Rysunek 3.6: Kod funkcji calcAverageAttractivness, która jest jedną z funkcji, pomagających dzikom znaleźć najlepsze pole do przemieszczenia się.

3.2.3 Point

Klasa Point reprezentuje jedno pole na mapie (czyli zgodnie z przyjętą konwencją teren o powierzchni 1ha). Przechowuje ona dane o sąsiadach (sąsiedztwo Moore'a) oraz pozwala na pojawianie się jedzenia (Rys.3.7).

```
public class Point{

    private static final int SFMAX = 100000;

    public ArrayList<Point> neighbors;

    @Deprecated
    public static Integer [] types = {0,1,2,3,4};// 0 - pole nieokreślone, 1 - las, 2 - podmokłe,
    // 3 - miasto, 4 - myśliwy

    public boolean garbageCollection;
    public int type;
    public int staticField;
    public int x;
    public int y;

    public void setCurrentFood(float currentFood) { this.currentFood = currentFood; }

    private float currentFood;
    private float foodCap;
    private Random rng = new Random();
    private static final float MIN_FOOD_PER_ROUND = 0.008f;
    private static final float MAX_FOOD_PER_ROUND = 0.025f;
    private static final float MIN_STARTING_FOOD = 0.05f;
    private static final float MAX_STARTING_FOOD = 1.0f;
    private static final float MIN_FOOD_CAP = 1.0f;
    public static final float MAX_FOOD_CAP = 3.0f;
    private static final float MIN_GARBAGE_FOOD = 10.0f;
    public static final float MAX_GARBAGE_FOOD = 25.0f;
```

Rysunek 3.7: Zmienne oraz stałe, przechowywane przez klasę Point.

3.2.4 Dzik

Klasa Dzik reprezentuje watahę dzików. Posiada ona informację o położeniu stada, liczbie członków oraz stałych, regulujących życie dzika (Rys.3.8).

Najważniejszyą metodą tej klasy jest funkcja move (Rys.3.9).

```java
public class Dzik {
    private int x;
    private int y;
    private final Board board;
    private int dziksHere;
    private float hungerLevel = 0.0f;
    //Current hunger level per Dzik. Dzik won't eat, if it's negative.
    // The higher the hunger, the more it affects Dzik's movement

    private static final float MOVING_DZIK_CONSUMPTION_RATE = 0.7f;
    //How much food per round does Dzik need if it is moving

    private static final float STATIC_DZIK_CONSUMPTION_RATE = 0.3f;
    //How much food per round does Dzik need if it stays on the same field

    private static final float DZIK_MAX_HUNGER = 50.0f;
    //Value at which Dzik dies

    private static final float DZIK_MIN_HUNGER = -10.0f;
    //How much dzik can eat "in advance"

    private static final float HUNGER_FACTOR_MULTIPLIER_A = 0.38f; //A multiplier in foodFactor = exp((hunger*A)*B
    private static final float HUNGER_FACTOR_MULTIPLIER_B = 10000.0f; //B multiplier in foodFactor = exp((hunger*A)*B

    private static final int ATTRACTIVENESS_RANDOMNESS = 20000;
```

Rysunek 3.8: Zmienne oraz stałe, przechowywane przez klasę Dzik.

```java
  public void move(){
      final float foodFactor = (float) (Math.exp(this.hungerLevel * HUNGER_FACTOR_MULTIPLIER_A)
              * HUNGER_FACTOR_MULTIPLIER_B);

      ArrayList<Point> neighbors;
      neighbors = (ArrayList<Point>) board.points[x][y].neighbors.clone();
      neighbors.add(board.points[x][y]);
      Collections.shuffle(neighbors);


      Point p = neighbors.stream()
              .filter(p3 -> p3.x>0 && p3.x <= Board.MAX_SIZE && p3.y>0 && p3.y <= Board.MAX_SIZE)
              .max(Comparator.comparingInt(
                      p2 -> p2.staticField
                      + (int) (p2.calculateFoodSmell(Board.DZIK_SENSE_RADIUS)*foodFactor)
                      + (int) (Math.random() * ATTRACTIVENESS_RANDOMNESS))
              ).get();
      board.dziks.get(board.points[x][y]).remove(this);

      if(p.x == x && p.y == y)
          this.hungerLevel += STATIC_DZIK_CONSUMPTION_RATE;
      else
          this.hungerLevel += MOVING_DZIK_CONSUMPTION_RATE;

      x = p.x;
      y = p.y;
      board.dziks.get(board.points[x][y]).add(this);
  }
```

Rysunek 3.9: Funkcja move, poruszająca watahą dzików.

3.2.5 IterationStatistics

Klasa IterationStatistics pomaga w zbieraniu danych do statystyk. Jej najważniejszą metodą jest funkcja zamieniająca dane na tekst, gotowy do zapisania w pliku (Rys.3.10).

```
public class IterationStatistics {
    private int iteration;
    private int dziks;
    private int dziksInCities;
    private int dziksInTheWild;
    private int food;

    public String toString() {
        return Integer.toString(iteration) + " " + Integer.toString(dziks) + " " + Integer.toString(dziksInCities) +
                " " + Integer.toString(dziksInTheWild) + " " + Integer.toString(food);
    }

}
```

Rysunek 3.10: Większość kodu klasy IterationStatistics.

3.2.6 GUI

Klasa GUI jest odpowiedzialna za oprawę graficzną programu. Jej kod prawie w całości został nam dostarczony przez prowadzących.

Rozdział 4

Przeprowadzone symulacje

4.1 Sposoby rozmieszczania myśliwych

Podczas laboratoriów przeprowadziliśmy ankietę, w której zapytaliśmy o możliwe najlepsze rozmieszcznie myśliwych na badanym obszarze w taki sposób, by skutecznie zminimalizować (ale nie wyzerować) liczbę dzików.

Ankietowani zazwyczaj rozmieszczali myśliwych według jednej z trzech strategii:

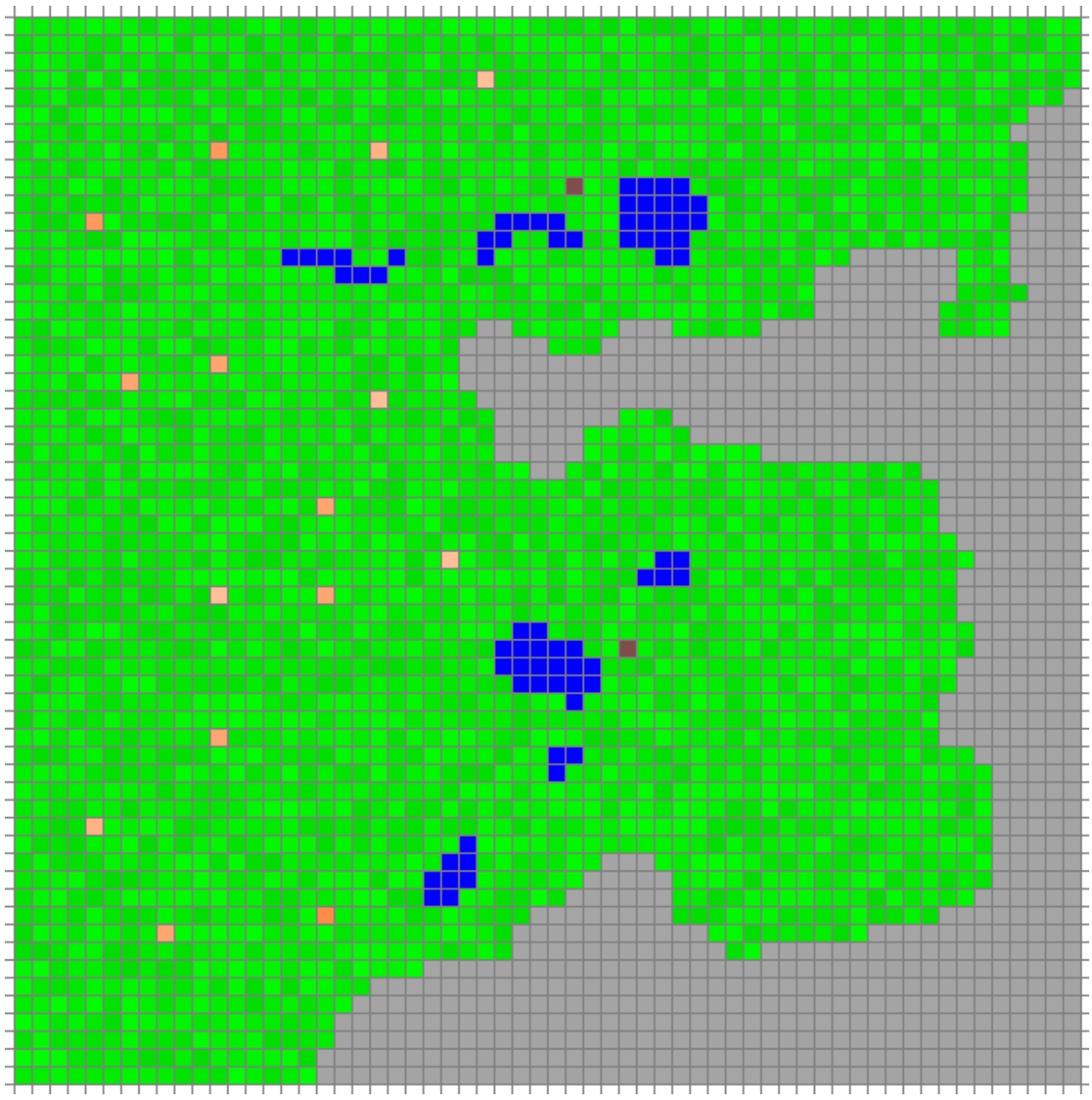
- w środku lasu
- zaraz nad brzegiem jezior
- na obrzeżach miast

Przeprowadziliśmy symlulacje z różnymi ustawieniami, inspirując się wynikami ankiety. Podczas przeprowadzania symulacji zbieraliśmy informacje o ogólnej liczbie dzików i liczbie dzików w mieście w danym momencie.

4.2 Wyniki przeprowadzonych symulacji

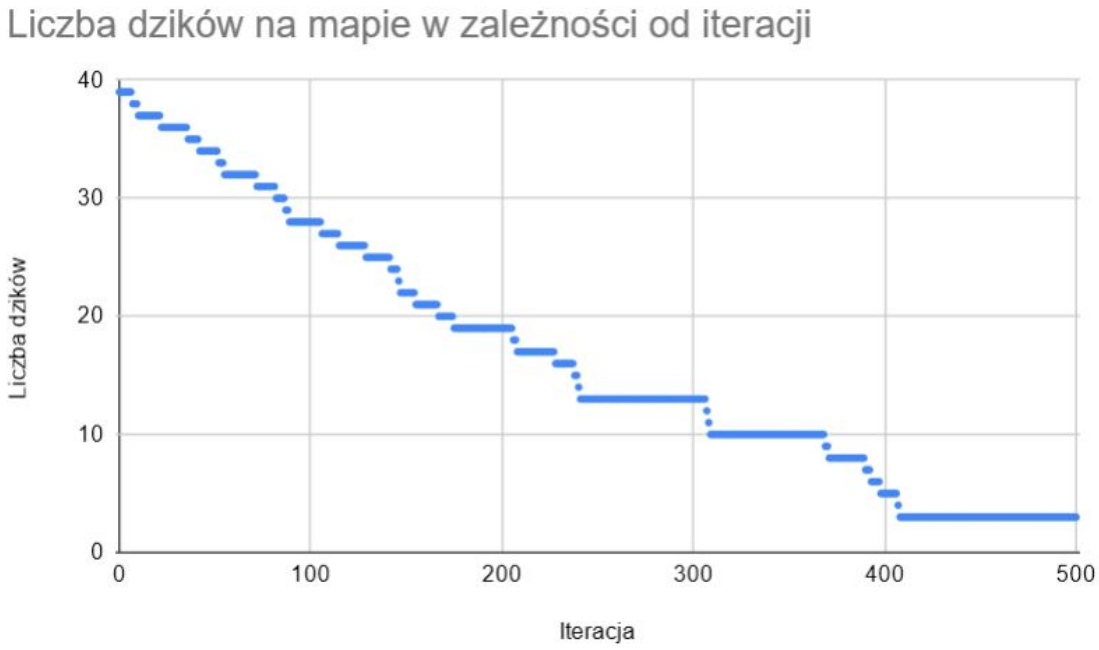
W sprawozdaniu zostaną przedstawione wyniki symulacji z rozmieszczeniem dwóch myśliwych na różne sposoby. Każda symulacja działała przez 1000 iteracji. Dla każdej symulacji początkowe rozstawienie dzików było pseudo losowe (pilnowaliśmy, żeby nie symulować niezbyt realistycznych scenariuszy, np. wszytskich dzików, zaczynających symulację zraz obok siebie).

- **Myśliwi nad brzegiem jezior**
  Rozstawienie myśliwych, na którym była prowadzona symulacja, widać na rysunku 4.1.

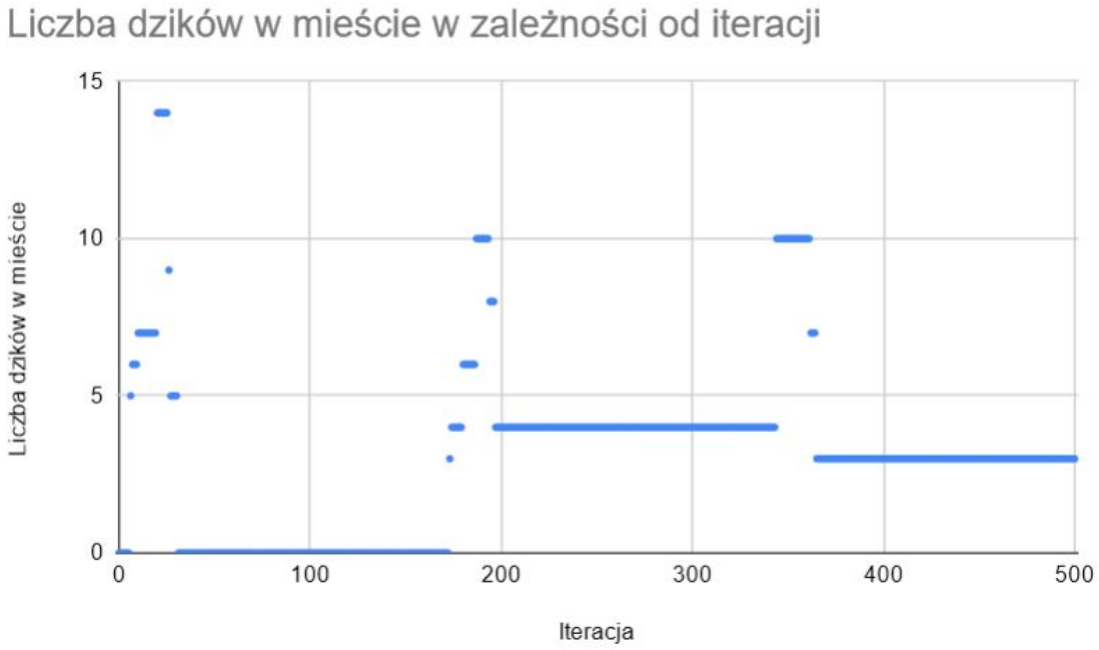


Rysunek 4.1: Mapa przed rozpoczęciem symulacji.

Liczba żywych dzików oraz liczba dzików w mieście zostały przedstawione na wykresach, widocznych odpowiednio na rysunkach 4.2 oraz 4.3. .



Rysunek 4.2: Wykres, prezentujący zebrane dane.



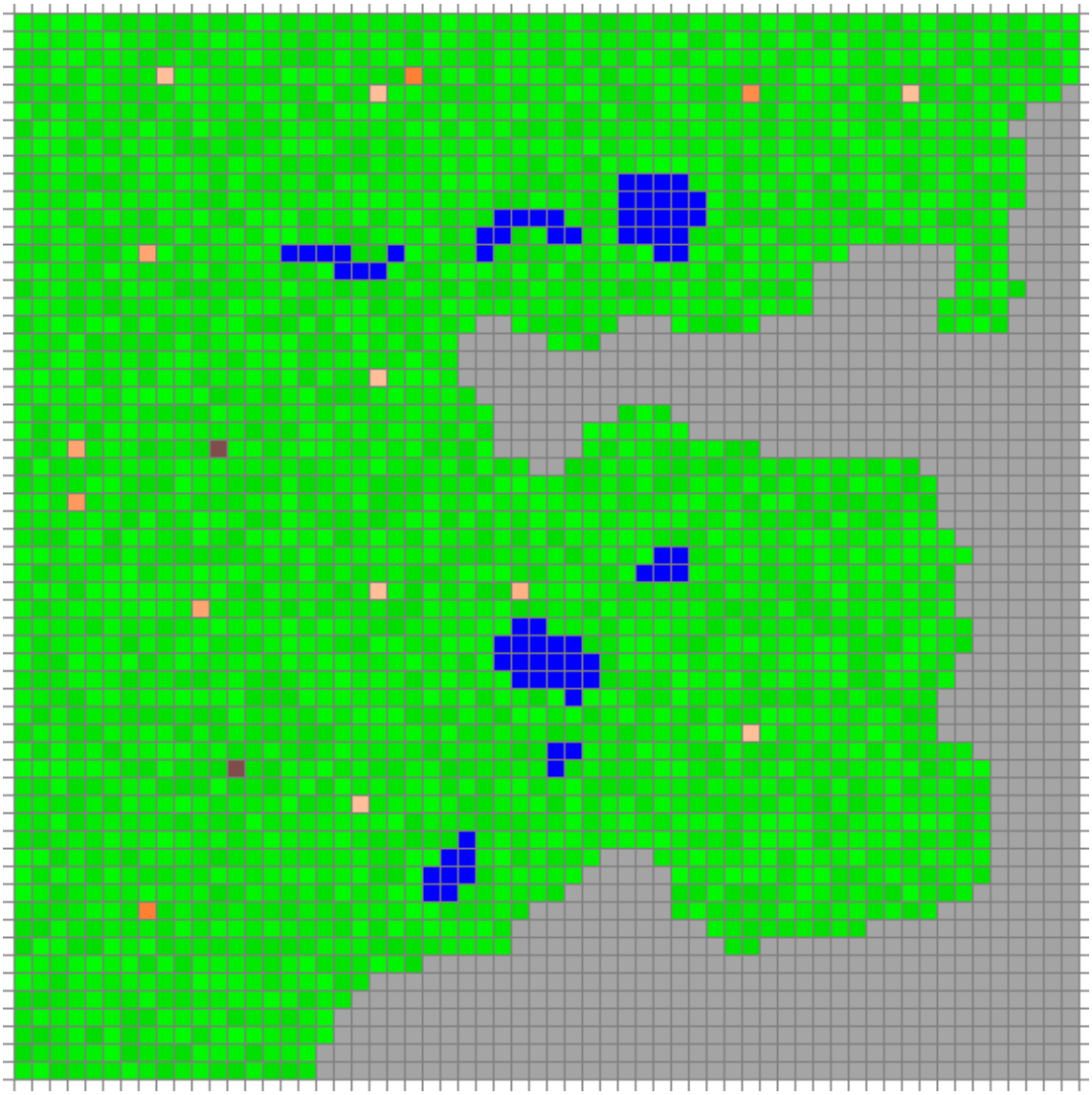
Rysunek 4.3: Wykres, prezentujący zebrane dane.

Obserwacje: To rozstawienie pozwalało systematycznie pozbywać się dzików,

ponieważ cały czas przychodziły one na atrakcyjne tereny podmokłe i nawet uwzględniając niską atrakcyjność pola, na którym zginął niedawno inny dzik, było ich tam bardzo dużo. Pojawił się jadnak problem, ponieważ watahy, zniechęcone terenem, na którym zastrzelono innego dzika, bardzo często (częściej niż przy innych rozstawieniach) wchodziły w głąb miasta i nie umiały znaleźć z niego wyjścia. Jest to sytuacja potencjalnie niebezpieczna.

- **Myśliwi w środku lasu**

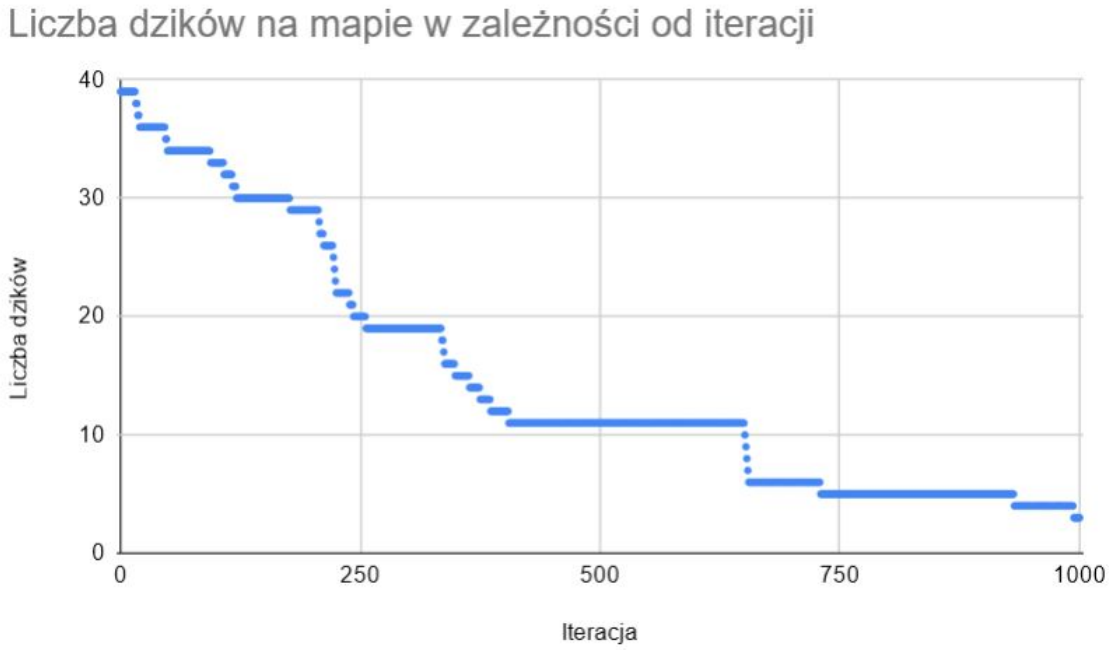
  Rozstawienie myśliwych, na którym była prowadzona symulacja, widać na rysunku 4.4.



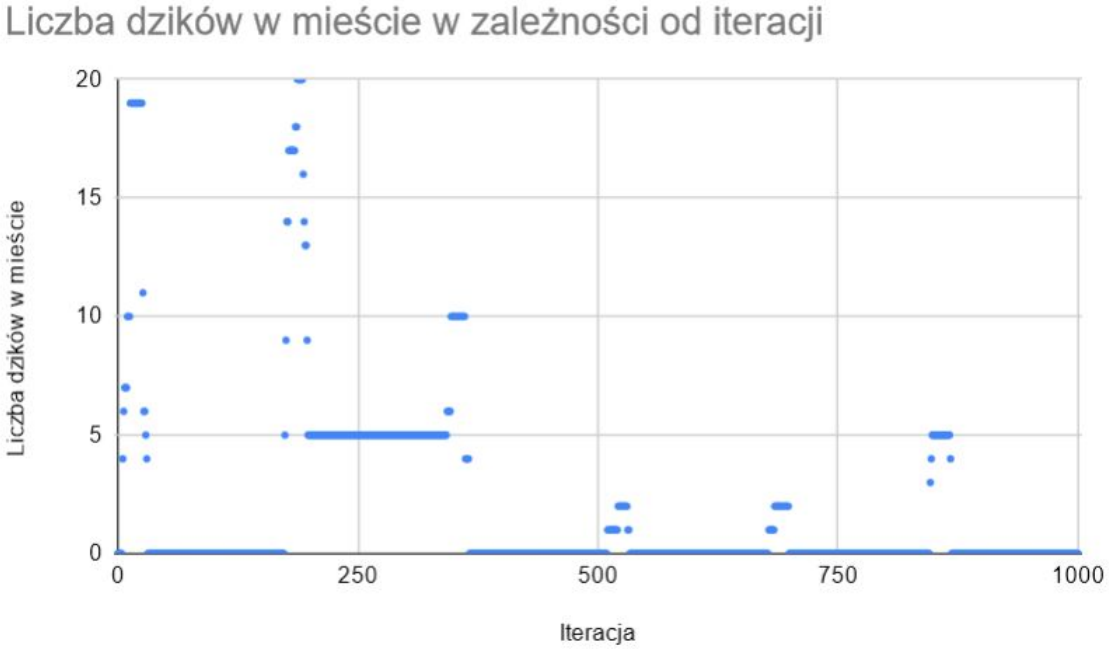
Rysunek 4.4: Mapa przed rozpoczęciem symulacji.

Liczba żywych dzików oraz liczba dzików w mieście zostały przedstawione na wykresach, widocznych odpowiednio na rysunkach 4.5 oraz 4.6.

Obserwacje: Ta metoda najwolniej pozbywała się dzików. Ma to swoje plusy, ponieważ dzięki temu nie było ryzyka wybicia całej populacji. Niestety podobnie jak w przypadku rozstawienia myśliwych przy terenach podmokłych, wystraszone dziki wędrowały często w stronę miast.



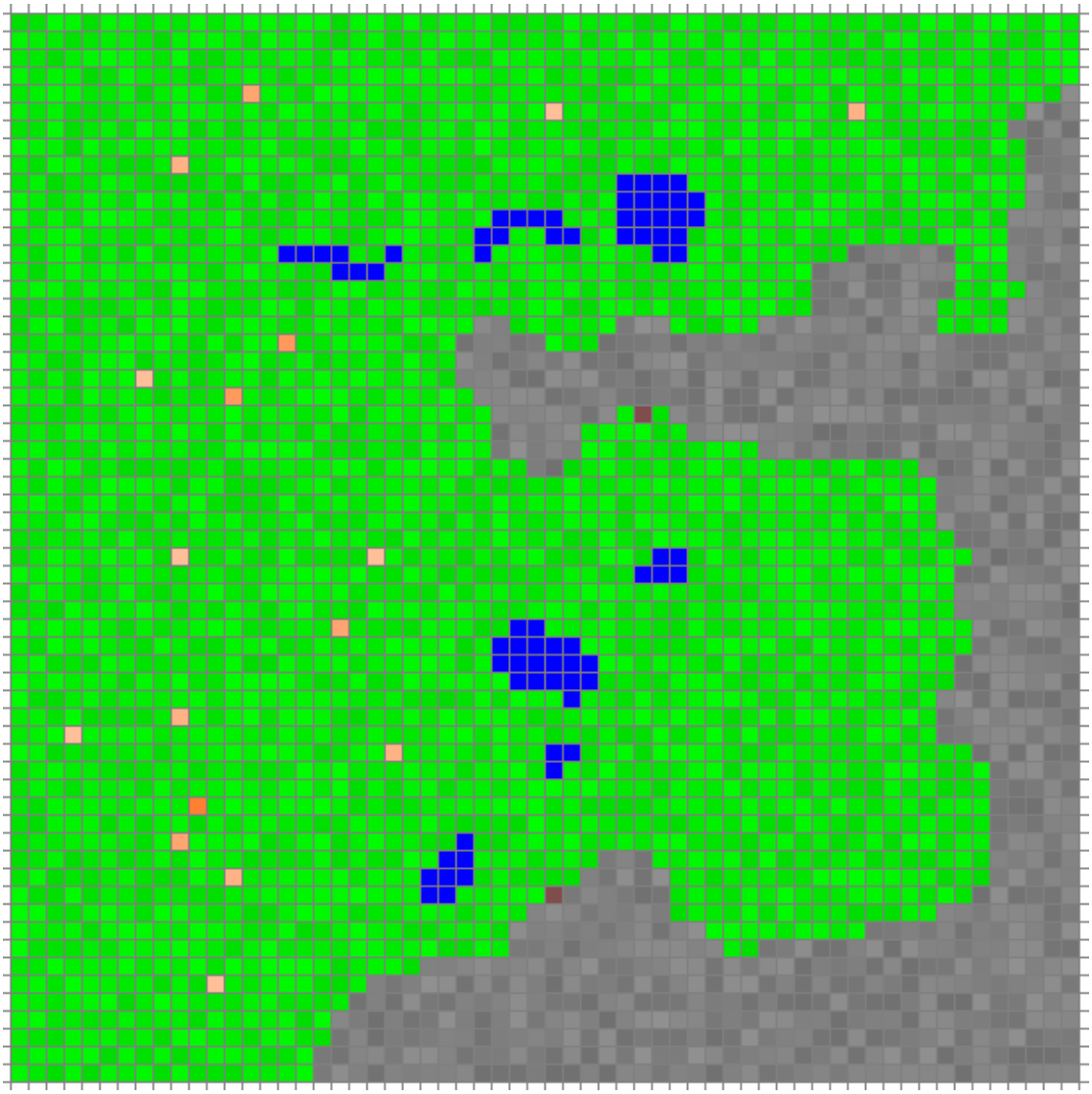
Rysunek 4.5: Wykres, prezentujący zebrane dane.



Rysunek 4.6: Wykres, prezentujący zebrane dane.

- **Myśliwi na obrzeżach miast**

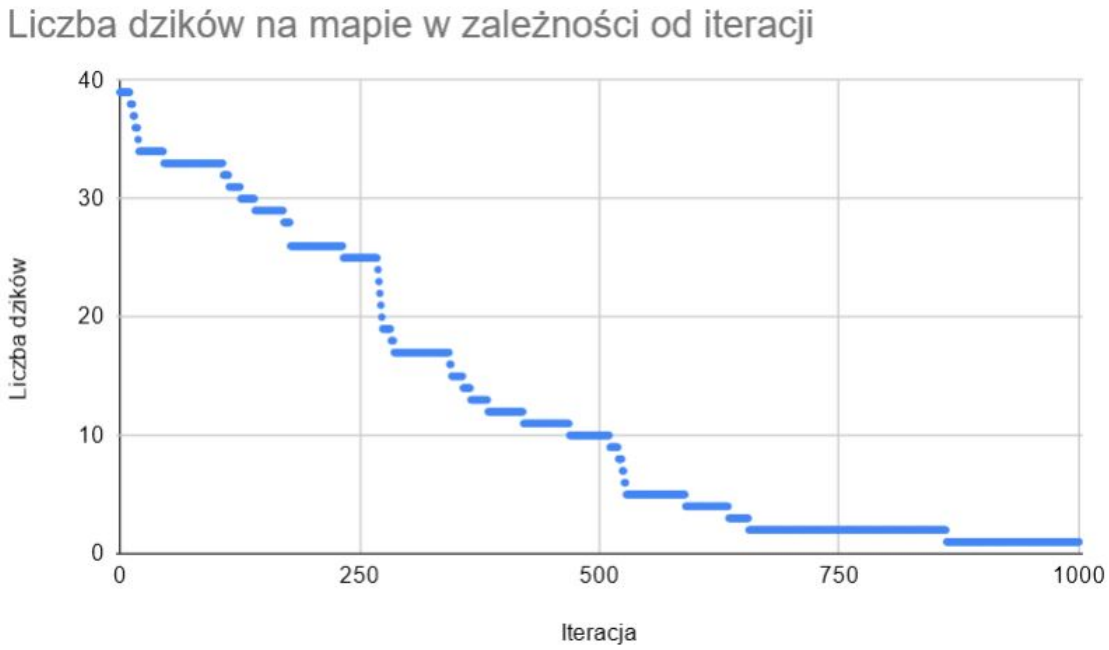
  Rozstawienie myśliwych, na którym była prowadzona symulacja, widać na rysunku 4.7.



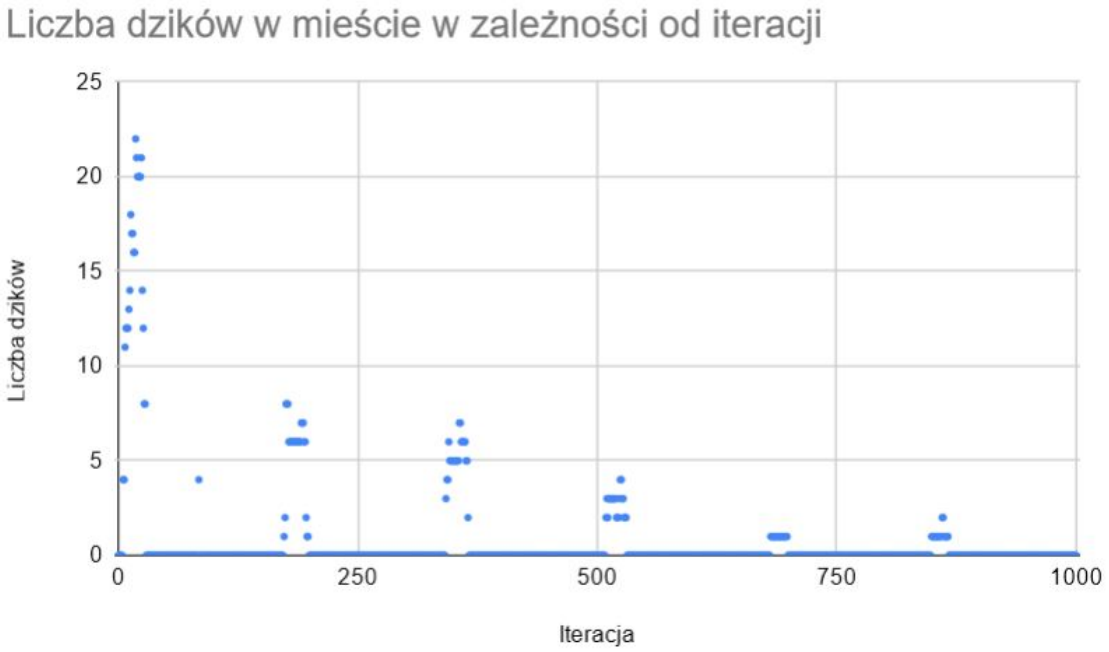
Rysunek 4.7: Mapa przed rozpoczęciem symulacji.

Liczba żywych dzików oraz liczba dzików w mieście zostały przedstawione na wykresach, widocznych odpowiednio na rysunkach 4.8 oraz 4.9.

Obserwacje: Przy tym rozstawieniu nie zaobserwowano dzików gubiących się na terenach miejskich. Metoda ta pozwalała w ekspresowym tempie pozbywać się dzików. Ich liczba malała najszybciej w dni wywozu śmieci, kiedy to podchodziły one do miast, na brzegu których czekali myśliwi. Jest to jednak o tyle niebezpieczne, że w niektórych symulacjach z podobnym rozstawieniem liczba dzików została wyzerowana, co zagraża ekosystemowi.



Rysunek 4.8: Wykres, prezentujący zebrane dane.



Rysunek 4.9: Wykres, prezentujący zebrane dane.

Rozdział 5

Wnioski

Pomimo wielu prób oraz nie udało nam się znaleźć zadowalającego rozstawienia. W każdym przypadku albo liczba dzików malała za szybko, albo znajdowały one sposób na przedostanie się do miast. Nie można oczywiście ignorować też kwestii prawnych- symulacje prowadzone z myśliwymi na brzegach miast, pomimo dobrych wyników nie będą mogły być odwzorowane w rzeczywistości, ponieważ mieszkańcy miasta mogli by się poczuć zagrożeni.

Niektóre rozwiązania pomagały zmiejszyć ogólną liczbę dzików, jednak przez to, że dziki uciekały od miejsca postrzału (często w stronę miast), niekoniecznie oznaczało to jednak zmniejszenie liczby dzików w miastach.

Trzeba wspomnieć o tym, że w naszej symulacji nie było limitu dzików, jakie jeden myśliwy mógł upolować. W rzeczywistości takie limity istnieją, jednak do tego liczba dzików w lasach stale musi być kontrolowana (nierzadko szacowana). Uwzględniając takie limity można by poprowadzić symulacje z większą liczbą myśliwych. Pojawia się wtedy jednak kolejny problem- skąd tych wszytkich myśliwych wziąć?...

Podsumowując, problem dzików w miastach jest naprawdę złożony i myślistwo nie jest dostateczną odpowiedzią. Żeby zapewnić mieszkańcom bezpieczeństwo potrzebne byłyby dodatkowe środki zaradcze.

Bibliografia

[1] Artykuł o dzikach w miastach w radiu eska. https://www.eska.pl/slaskie/dziki-upodobaly-sobie-miejskie-osiedla-aa-eHb2-WWJr-U8U8.html. Dostęp: 5.07.2023r.

[2] Google maps. https://www.google.pl/maps/place/Ligota,+40-710+Katowice/@50.2191986,18.9567296,15z/data=!4m6!3m5!1s0x4716cebfcc951deb:0x2738fc7ccfcf884d!8m2!3d50.2226496!4d18.974942!16s%2Fg%2F122h6qc3?entry=ttu. Dostęp: 4.07.2023r.

[3] Raport o postrzeleniu człowieka przez myśliwego. https://tvn24.pl/polska/zawadka-rymanowska-mysliwy-pomylil-na-polowaniu-kolege-z-dzikiem-postrzelil-g
Dostęp: 5.07.2023r.

[4] Raport telewizji tvs o porodzie dzików w mieście. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwii_Pzo8_f_AhXSFXcKHYLJBlMQFnoECAoQAQ&url=https%3A%2F%2Ftvs.pl%2Finformacje%2Fkatowice-porod-w-ogrodku-na-ligocie-zdjecia-na-swiat-przyszly-malutkie-dzik
2F&usg=AOvVaw3JdorJYQAK2JxTmZ7Mwktm&opi=89978449. Dostęp: 4.07.2023r.

[5] Strona edukacyjna okręgów łowieckich. https://www.pzlow.pl/edukacja. Dostęp:5.07.2023r.

[6] Strona edukacyjna policji warmińsko-mazurskiej. https://warminsko-mazurska.policja.gov.pl/ol/aktualnosci/7833,Elblag-Dzik-jest-dziki-ale-nie-jest-zly-Jak-sie-zachowac-gdy-spotkamy-na-osie
ht. Dostęp:5.07.2023r.