WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

Symulacje Komputerowe

Sprawozdanie "Symulacja Wieloagentowa"

Adam Talarczyk, Mateusz Wrzoł

Uniwersytet Śląski, Sosnowiec, 2021

Spis treści

1 Zadanie 1

Należy opracować symulator dowolnego zjawiska lub procesu, wykorzystując model wieloagentowy.

Symulator powinien być wyposażony następujące funkcje:

- wizualizacja stanu środowiska i agentów,
- wykres(y) z wynikami symulacji,
- interfejs użytkownika umożliwiający modyfikowanie parametrów modelu.

Sprawozdanie powinno zawierać:

- opis zaimplementowanego modelu wieloagentowego,
- kod źródłowy symulatora z komentarzami,
- prezentację interfejsu użytkownika z zrzutami ekranu,
- przykładowe wyniki symulacji,
- spis bibliografii (jeżeli była wykorzystana).

Dodatkowo poza sprawozdaniem proszę przesłać pik(i) z projektem symulatora (np. plik Netlogo).

Ocena rozwiązania będzie uwzględniała:

- stopień skomplikowania zaproponowanego modelu i opracowanego symulatora,
- oryginalność rozwiązania (symulator nie może być prostą modyfikacją modeli symulacyjnych dostępnych w Netlogo lub innego gotowego oprogramowania),
- jakość przygotowanego sprawozdania.

Do rozwiązania zadania można wykorzystać środowisko Netlogo lub dowolne inne środowisko programistyczne.

2 Symulacja środowiska ptaszników

Wykonana w sprawozdaniu symulacja jest odwzorowaniem naturalnego środiwiska ptaszników z uwzględnieniem ich charakterystycznych zachowań. Ponadto, symulacja pozwala na zmianę wielu czynników wpływając na zachowania symulowanych obiektów.

Rozdział zawiera opis zaimplementowanego modelu wieloagentowego, kod źródłowy z komentarzami, prezentację interfejsu użytkownika oraz opis wyników symulacji.

2.1 Opis modelu

Stworzony model obejmuje zagadnienia związane z przebiegiem cyklu życia ptasznika, lecz bez wskazania na konkretny gatunek. Użytkownik posiada możliwość zasymulowania według własnego zamysłu grupowej hodowli ptaszników. Projekt został zaimplementowany na podstawie poniższych zagadnień:

- Ptaszniki podzielone są na dwie płcie: samiec oraz samica,
- Przestrzeń, po której poruszają się ptaszniki zawiera miejsca wilgotne, a także suche,
- W miejscach wilgotnych samice oraz podrostki samców zakładają gniazda, mogę jednak je opuścić w poszukiwaniu innej lokalizacji,
- Na całym obszarze poruszają się owady karmowe, które służą jako pokarm dla populacji ptaszników, ich ilość jest regulowana,
- Mijający czas powoduje wzrost ptaszników odpowiednio wielkościowo dla płci (pajęczaki przechodzą tak zwaną wylinkę), a także powoduje śmierć u zwierząt ze względu na podeszły wiek,
- Okres oczekiwania na kolejną wylinkę ptasznika może być regulowany przez użytkownika systemu,
- Spotkanie dwóch ptaszników może doprowadzić do walki pomiędzy nimi, a w następstwie śmiercią jednego z nich,
- Dorosłe samce poruszają się po terytorium w poszukiwaniu samicy, w celu odbycia kopulacji,
- Kopulacja może skończyć się pozytywnym scenariuszem, wtedy samica po pewnym czasie może złożyć kokon. W najgorszym wypadku samica atakuje samca i go zjada, dzieje się tak w momencie gdy samica jest głodna lub wskaźnik agresji wśród populacji znajduje się na wysokim poziomie,

- Udana kopulacja nie zapewnia pewności stworzenia kokonu przez samicę, powodzenie gwarantuje odpowiedni parametr ustawiany przez użytkownika,
- Dojrzałość płciową reguluje się w systemie za pomocą parametru DC, w żargonie terrarystyki jest to długość ciała zwierzęcia,
- Złożenie kokonu odbiera samicy połowę maksymalnej energii, a opieka nad kokonem samica sprawuje do czasu jego otwarcia,
- Inkubacja kokonu trwa przez okres zdefiniowany przez użytkownika,
- Z otwierającego się kokonu wychodzi określona przez użytkownika ilość nimf (okres życia ptasznika po wylęgu),
- Warunki atmosferyczne (temperatura i wilgotność) mają wpływ na życie ptaszników oraz szansę otwarcia się kokonu, niekorzystne warunki powodują wymieranie populacji, Wskaźniki numeryczne oraz wykresy wskazują parametry związane z ilością poszczególnych elementów w symulacji oraz ich zmiany w przeciągu interwałów czasowych.

2.2 Kod źródłowy symulatora- wybrane fragmenty

W tym rozdziale zostaną zaprezentowane kluczowe elementy kodu źródłowego symulacji. Podstawowym elementem symulacji są agenci, których implementacja jest zawarta w listingu 1.

```
breed [spiders spider]
spiders-own [ sex dc energy nest health last-molt]
```

Listing 1: Implementacja agenta w środowisku NetLogo

W linii pierwszej definiuje się agenta typu “spiders”, zaś linia druga określa parametry z nim związane.

W sekcji “setup” ustawiane są wartości związane z podstawowymi parametrami symulacji, szczegóły zostały zaprezentowane w listingu 2.

```
to setup
  clear-all
  reset-ticks
  set-default-shape spiders "spider"
  set-default-shape bugs "bug"
  set-default-shape cocoons "egg"
  set maxsize-female 1.1
  set maxsize-male 0.95
  set-background
  spawn-spiders
end
```

Listing 2: Ustalanie podstawowych parametrów symulacji.

W linii 2 oraz 3 usuwane są wartości pochodzące z poprzedniego użycia symulacji, następnie (linia 4, 5, 6, 7 i 8) zostają ustalone kształty jakie zostają nadane poszczególnym agentom oraz nadania im maksymalnego rozmiaru. Stworzenie tła na podstawie zdefiniowanej procedury znajduje się w linii 9, zaś pojawienie się ptaszników uzależnione jest od wywołania w linii 10.

Zdefiniowanie podstawowych zmiennych związanych z ptasznikiem zawarte jest w listingu 3, także tutaj znajduje się w linii 1 stworzenie według parametru, określonej liczby ptaszników:

```
1  create-spiders ilosc_samic [
2      set color pink
3      setxy random-xcor random-ycor
4      set energy 100
5      set health 100
6      set sex 1
7      set nest false
8      set dc max_dc
9      set last-molt ticks
10     set size maxsize-female
11   ]
```

Listing 3: Metoda odpowiadająca za tworzenie obiektu typu "ptasznik samica".

Wykonywanie się poszczególnych funkcji jest uzależnione od zawarcia jej w sekcji "go". W tym miejscy wykonuje się dla każdego z agentów określona procedura, która musi zostać do niego przypisana. W listingu 4 został pokazany przykład na podstawie symulacji.

```
1  to go
2    if not any? spiders [ stop ]
3    spawn-food
4    ask spiders
5    [ move
6      eat
7      will-fight
8      reproduce
9      make-nest
10     molting
11     male-wait-to-grow-up
12     random-disaster
13     death ]
14
15   ask bugs
16   [ move-food
17     death-food ]
18
19   ask cocoons
20   [ hatch-cocoon ]
21   tick
```

```
22 end
```

Listing 4: Sekcja “go” symulacji.

Sekcja “go” (linia od 1 do 20) wykonuje się w momencie istnienia przynajmniej jednego ptasznika, potwierdza to kod znajdujący się w linii 2. W danym ticku (jednostka miary symulacji) wykonuje się dla obiektów symulacji szereg procedur (na podstawie ptasznika od linii 4 do 13), następnie proces przechodzi do kolejnego ticku (linia 19) i tym samym się zapętla przechodząc ponownie do linii 1.

Zachowanie agenta jest uzależnione od zaimplementowanej funkcji wywoływanej w procedurze “go”. Przykładowa zawartość takiego elementu znajduje się w listingu 5.

```
1  to reproduce
2    let candidate one-of spiders-at 1 0
3    if candidate != nobody [
4      if (([sex] of candidate) = 1 and sex = 0 and
5        ([nest] of candidate) = true) [
6        if ([energy] of candidate < 50)[
7          set health 0
8          ask candidate [ set energy 100 ]
9        ]
10       if([dc] of candidate) >= dc_kopulacja and dc >=
     dc_kopulacja and health != 0[
11         create-cocoon
12       ]
13     ]
14   ]
15 end
```

Listing 5: Przykładowa procedura “reproduce”

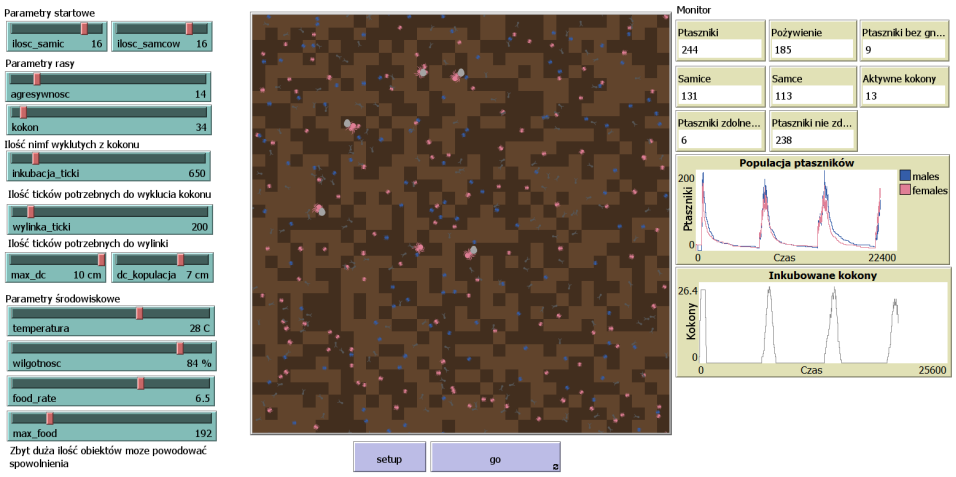
Linia 2 ładuje do zmiennej “candidate“ informacje na temat napotkanego agenta tego samego typu- spiders. W momencie gdy obiekt nie jest pusty (linia 3), sprawdza się warunek, czy dany ptasznik jest samcem a spotkany obiekt samicą, ulokowaną w swoim gnieździe (linia 4 i 5). Wartość “true” obliguje do weryfikacji parametru odpowiadającego za energię samicy, wartość mniejsza od 50 powoduje, że samiec zostaje zjedzony przez samicę, zyskując maksymalną ilość energii. W przeciwnym wypadku porównywana jest minimalna wielkość ptasznika zdolnego do kopulacji. Spełnienie warunku wywołuje procedurę “create-cocon”, odpowiedzialną za utworzenie kokonu przez samicę (linia 11).

2.3 Interfejs użytkownika

Interfejs użytkownika został przedstawiony na rysunku 1, wyszczególnić można 3 kategorie:

- Konfiguracja symulacji i środowiska

- Pole na którym wyświetlana jest symulacja
- Panel raportowania, zawierający dane i wykresy



Rysunek 1: Obszar symulacji

Obszar konfiguracji środowiska również podzielony jest na kategorie. Parametry startowe zaprezentowane na rysunku 2 (tj. *ilosc_ samic*, *ilosc_ samcow*) określają ile dorosłych samic i samców podczas inicjalizacji ma zostać wygenerowanych.

Rysunek 2: Kontrolki odpowiedzialne za parametry startowe

Parametry rasy (rysunek 3 określają ogólne zachowanie ptaszników:

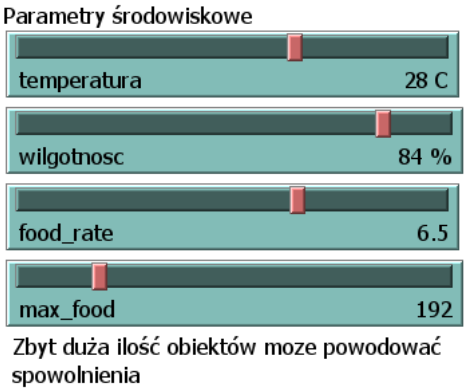
- *agresywnosc* - zachowanie w stosunku do innych ptaszników. Wartość 0 oznacza, że nie będąze sobą w ogóle walczyć,
- wartość 100 – walka przy każdym spotkaniu.
- *kokon* - ilość młodych nimf wyklutych z pojedynczego kokonu
- *inkubacja_ ticki* - ilość czasu potrzebne do wyinkubowania kokonu
- *wylinka_ ticki* - ilość czasu, potrzebna do odbycia kolejnej wylinki. Wylinka wiąże się ze wzrostem DC pająka.
- *max_ dc* – maksymalny rozmiar, który może osiągnąć ptasznik.

- *dc_kopulacja* – wymagany rozmiar ptasznika aby osiągnąć zdolności kopulacyjne

Rysunek 3: Kontrolki odpowiedzialne za parametry rasy

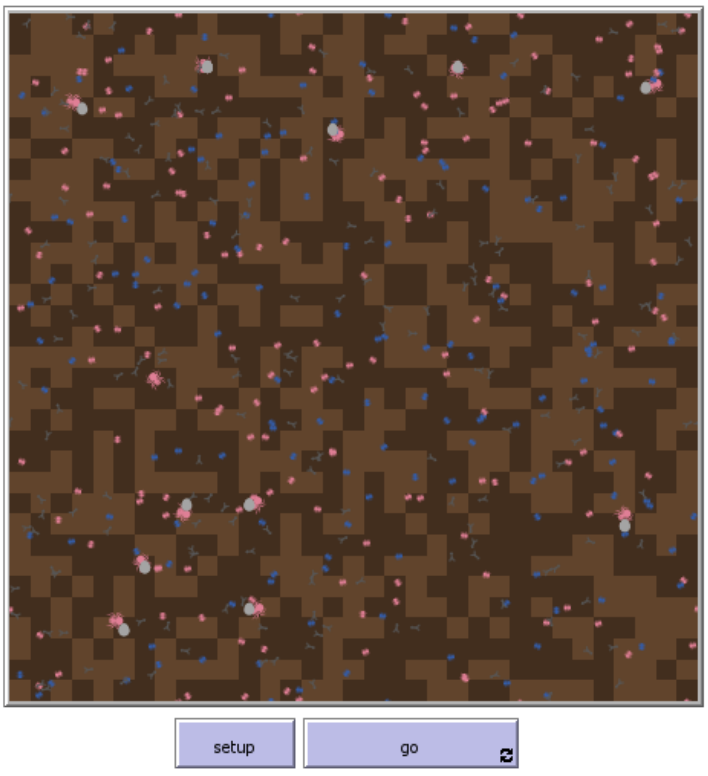
Parametry środowiskowe - ogólne parametry odpowiedzialne za środowisko, temperaturę, wilgotność oraz ilość pożywienia na ekranie (rysunek 4):

- *temperatura*, *wilgotnosc* - parametry, które mogą być zmieniane podczas symulacji. Stosując wartości skrajne można spowodować zachowania uboczne, np. Umieranie ptaszników, niszczenie kokonów.
- *food_rate* - mnożnik jedzenia, im większy, tym więcej pożywienia pojawia się na ekranie.
- *max_food* - maksymalna ilość pożywienia na ekranie jednocześnie. Zbyt duża ilość może powodować problemy optymalizacyjne.

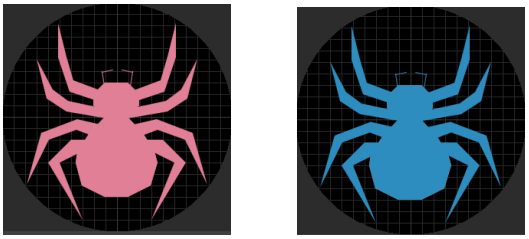


Rysunek 4: Kontrolki odpowiedzialne za parametry środowiskowe

Obszar symulacji zaprezentowano na rysunku 5. Po obszarze poruszają się ptaszniki, których płeć można rozróżnić przez charakterystyczne kolory. Kolor różowy oznacza samicę, a niebieski samca (rysunek 6). Dodatkowo, na obszarze symulacji wyszczególnić można ikony kokonu i pożywienia (rysunek 7).

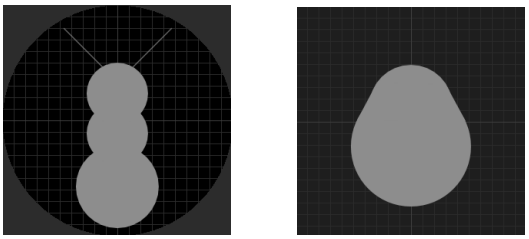


Rysunek 5: Obszar wizualizacji symulacji



(a) Ikona samicy (b) Ikona samca

Rysunek 6: Ikony samców i samic w symulacji



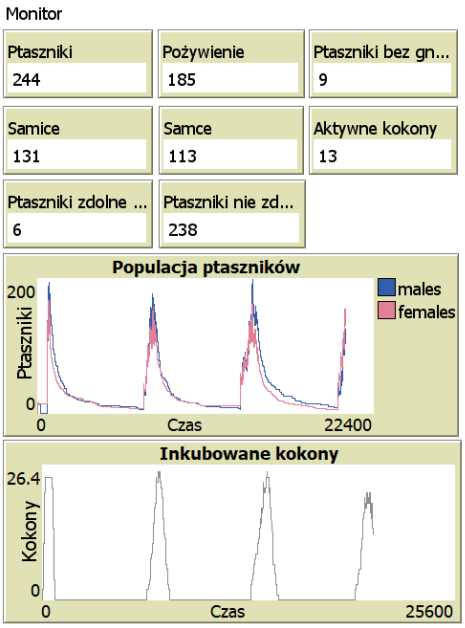
(a) Ikona pokarmu (b) Ikona kokonu

Rysunek 7: Pozostałe ikony użyte w symulacji

Dodatkowo, środowisko NetLogo pozwala na śledzenie obiektów w czasie rzeczywistym, zaprezentowane zostało to na rysunku 8

Rysunek 8: Parametry ptasznika w trakcie trwania symulacji.

Ostatnim elementem jest prezentacja danych i wyników. Odpowiedzialne jest za to monitor (rysunek 9).

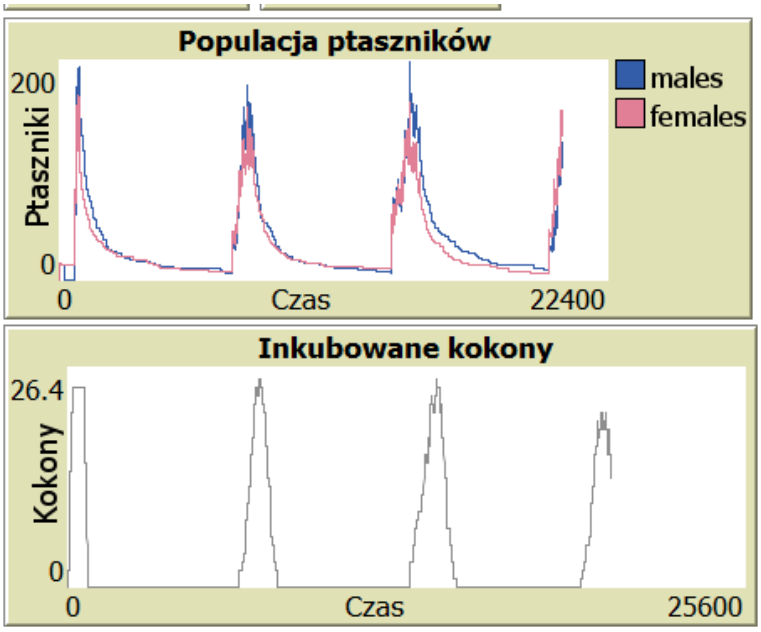


Rysunek 9: Monitor - reprezentacja danych symulacji

Monitor podzielony został on na dane i wykresy. Dane przedstawiają sumaryczna ilość ptasznikow z podziałem na samce, samice, ptaszniki zdolne i niezdolne do kopulacji. Dodatkowo, wyświetlana jest ilość pożywienia, ilość ptaszników bez gniazda oraz aktywne, inkubowane kokony (rysunek 10).

Rysunek 10: Poszczególne dane

Wykresy przedstawiają w czasie rzeczywistym populację ptaszników z podziałem na samce i samice. Ostatni wykres przedstawia ilość kokonów (rysunek 11).



Rysunek 11: Wykresy

2.4 Wyniki symulacji

Wyniki symulacji przedstawiane są w czesie rzeczywistym na monitorze danych. Wizualna reprezentacja przedstawiona jest na wykresach (rysunek 11). Zauważyć można zależność, że populacja ptaszników rośnie i maleje co jakiś interwał. Wynika to z charakterystyki ich zachowania, samce żyją znacznie krócej od samic i umierają niedługo po wylince umożliwiającej im kopulację. Ilość ptaszników, które przeżywa zależna jest od ilości pożywienia, agresywności rasy i innych przypadkowych czynników. Podsumowując, tylko niewielka ilość ptaszników dożywa momentu, w którym jest zdolna do kopulacji. Mimo to, w symulacji występuje ciągłość- gatunek nie wymiera przy optymalnych parametrach.

2.5 Kod źródłowy- pełny

```
breed [spiders spider]
spiders-own [ sex dc energy nest health last-molt]

breed [bugs bug]
bugs-own [ energy-value energy ]

breed [cocoons cocoon]
cocoons-own [ tick-created ]

globals [ maxsize-female maxsize-male ]
```

```
11
12 ;;Setup i inicjalizacja
13 to setup
14   clear-all
15   reset-ticks
16   set-default-shape spiders "spider"
17   set-default-shape bugs "bug"
18   set-default-shape cocoons "egg"
19   set maxsize-female 1.1
20   set maxsize-male 0.95
21   set-background
22   spawn-spiders
23 end
24
25 to set-background
26   ask patches [
27     if random-float 1000 < 500 [ set pcolor 31 ]
28     ifelse random-float 1000 > 500 [ set pcolor 32 ]
29     [ set pcolor 33 ]
30   ]
31 end
32
33 to spawn-spiders
34   create-spiders ilosc_samic [
35     set color pink
36     setxy random-xcor random-ycor
37     set energy 100
38     set health 100
39     set sex 1
40     set nest false
41     set dc max_dc
42     set last-molt ticks
43     set size maxsize-female
44   ]
45   create-spiders ilosc_samcow [
46     set color blue
47     setxy random-xcor random-ycor
48     set energy 100
49     set health 100
50     set sex 0
51     set nest false
52     set dc max_dc
53     set last-molt ticks
54     set size maxsize-male
55   ]
56 end
57
58
59 ;;Ptaszniki - zachowanie
```

```
to go
  if not any? spiders [ stop ]
  spawn-food
  ask spiders
  [ move
    eat
    will-fight
    reproduce
    make-nest
    molting
    male-wait-to-grow-up
    random-disaster
    death ]

  ask bugs
  [ move-food
    death-food ]

  ask cocoons
  [ hatch-cocoon ]
  tick
end

to move
  ifelse nest [
    set energy energy - 0.1
  ][
    rt random 50
    lt random 50
    fd 1
    set energy energy - 0.5
  ]
end

;Walka
to will-fight
  let candidate one-of spiders-at 1 0
  if candidate != nobody [
    if random 100 < agresywnosc [
      ifelse([dc] of candidate) <= dc_kopulacja and dc <=
    dc_kopulacja[
        fight
      ][
        if([sex] of candidate) = sex [ fight ]
      ]
    ]
  ]
end

```

```
to fight
  let candidate one-of spiders-at 1 0
  if candidate != nobody [
    ifelse([dc] of candidate) > dc [
      set health 0
    ]
    [
      ask candidate [ set health 0 ]
    ]
  ]
end

to eat
  let candidate one-of bugs-at 1 0
  if candidate != nobody [
    if(energy < 100) [set energy energy + ([energy-value] of
    candidate)]
  ]
end

to reproduce
  let candidate one-of spiders-at 1 0
  if candidate != nobody [
    if (([sex] of candidate) = 1 and sex = 0 and
      ([nest] of candidate) = true) [
      if ([energy] of candidate < 50)[
        set health 0
        ask candidate [ set energy 100 ]
      ]
      if([dc] of candidate) >= dc_kopulacja and dc >=
   dc_kopulacja and health != 0[
        create-cocoon
      ]
    ]
  ]
end

to create-cocoon
  let candidate one-of spiders-at 1 0
  hatch-cocoons 1 [
    set color 6
    set size 1
    set tick-created ticks
    setxy [pxcor] of candidate [pycor] of candidate
  ]
  ask candidate [ set energy energy / 1.5]
end

to hatch-cocoon
```

```
155   ask cocoons [
156     if ticks > tick-created + inkubacja_ticki [
157       ifelse( temperatura > 20 and wilgotnosc > 50) [
158         hatch-spiders kokon [
159           setxy random-xcor random-ycor
160           set energy 30
161           set health 100
162           set sex random 2
163           ifelse sex = 1 [set color pink][set color blue]
164           set nest false
165           set dc 1
166           set size 0.5
167           set last-molt ticks
168         ]
169       ][die]
170 
171       die
172     ]
173   ]
174 end
175 
176 to molting
177   ask spiders [
178     if ticks > last-molt + wylinka_ticki [
179       set last-molt ticks
180 
181       ifelse(sex = 0)[
182         ;male
183         ifelse(dc < max_dc) [set dc dc * 1.07][die]
184         if(size < maxsize-male) [set size size * 1.07]
185       ][
186         ;female
187         if(dc < max_dc) [set dc dc * 1.1]
188         if(size < maxsize-female) [set size size * 1.1]
189       ]
190       if( random 10000 > 9980 ) [die]
191 
192     ]
193   ]
194 end
195 
196 to death
197   if energy < 0 [ die ]
198   if health <= 0 [ die ]
199 end
200 
201 to make-nest
202   if (pcolor = 32 and sex = 1 and random 1000 > 900) [
203     set nest true
```

```
204   ]
205 end
206
207 to random-disaster
208   let candidate one-of cocoons-at 0 0
209   if (random 1000 > 995 and sex = 1 and candidate = nobody)
       [set nest false]
210   if (random 1000 > 995 and sex = 0) [set nest false]
211 end
212
213
214 to male-wait-to-grow-up
215   if (sex = 0) [
216     ifelse dc < dc_kopulacja [
217       set nest true
218     ][
219       set nest false
220     ]
221   ]
222 end
223
224 ;;Pozywienie - zachowanie
225 to spawn-food
226   if random-float 10 < food_rate [
227     if count bugs < max_food [
228       create-bugs 1 [
229         set color 3
230         set size 0.5
231         set energy-value random 20
232         set energy random 50
233         setxy random-xcor random-ycor
234       ]
235     ]
236   ]
237 end
238
239 to move-food
240   rt random 50
241   lt random 50
242   fd 1
243   set energy energy - 0.1
244 end
245
246 to death-food
247   if energy < 0 [ die ]
248 end
249
250 ;;Dane i statystyki
251 to-report count-males
```

```
252   let counter 0
253   ask spiders [
254     if sex = 0 [set counter counter + 1]
255   ]
256   report counter
257 end
258
259 to-report count-females
260   let counter 0
261   ask spiders [
262     if sex = 1 [set counter counter + 1]
263   ]
264   report counter
265 end
266
267 to-report count-can-copulate
268   let counter 0
269   ask spiders [
270     if (dc >= dc_kopulacja) [set counter counter + 1]
271   ]
272   report counter
273 end
274
275 to-report count-cannot-copulate
276   let counter 0
277   ask spiders [
278     if (dc < dc_kopulacja) [set counter counter + 1]
279   ]
280   report counter
281 end
282
283 to-report count-spiders-without-nest
284   let counter 0
285   ask spiders [
286     if (nest = false) [set counter counter + 1]
287   ]
288   report counter
289 end
```

Listing 6: Pełny kod źródłowy

Spis rysunków

Listings

Literatura

[1] https://ccl.northwestern.edu/netlogo/docs/programming.html

[2] https://ccl.northwestern.edu/netlogo/docs/dictionary.html