PROJEKT: SYMULACJA ROZNOSZENIA CHORÓB ZAKAŹNYCH W TŁUMIE

KOD KURSU: 120-ISI-1S-777
PRZEDMIOT: SYMULACJA SYSTEMÓW DYSKRETNYCH (SSD)

Autorzy projektu

Katarzyna Stępień

Bartłomiej Stępniewski

Mateusz Piękoś

Opiekun projektu

Dr inż. MARCIN PIEKARCZYK

EAIiIB / Katedra Informatyki Stosowanej

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Kraków, Polska

20 stycznia 2024 r.

Spis treści

1. CEL I ZAKRES PROJEKTU

1.1 Cel projektu

Projekt ten koncentruje się na opracowaniu symulacji rozprzestrzeniania się chorób zakaźnych, wykorzystując model automatu komórkowego do analizy zachowań agentów - reprezentujących ludzi w populacji. Głównym celem jest zrozumienie i przewidywanie dynamiki epidemii poprzez uwzględnienie różnych czynników, takich jak ruch agentów, ich interakcje oraz parametry związane z zakażeniem.

Symulacja pozwala na eksplorację wpływu różnych scenariuszy i strategii interwencyjnych, które mogą zastosowane w celu efektywniejszego zarządzania sytuacjami epidemicznymi. Projekt ten oferuje narzędzia do lepszego zrozumienia natury roznoszenia chorób zakaźnych w tłumie.

1.2 Oczekiwane rezultaty

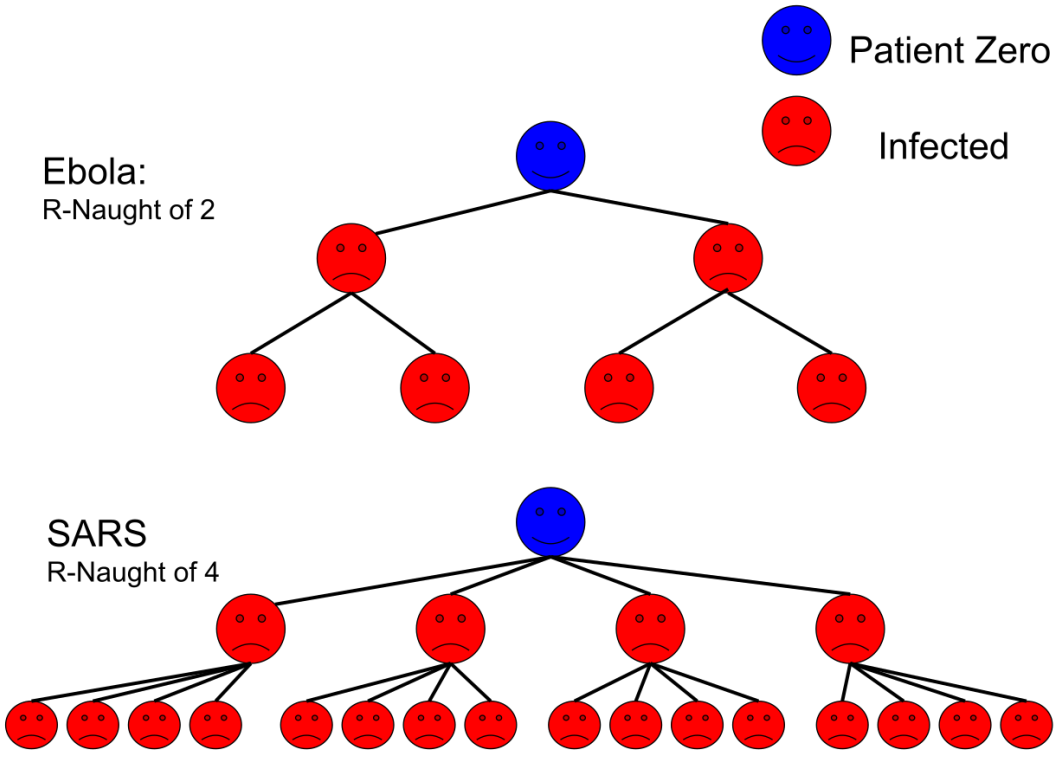
Nasz projekt zakłada stworzenie prostego narzędzia pozwalającego na symulowanie choroby o zadanych parametrach zaraźliwości na dostosowywanej przez użytkownika mapie. Symulacja ma obrazować możliwie realistyczne rozprzestrzenianie choroby zakaźnej, głównie w celach edukacyjnych.

2. CHARAKTERYSTYKA PROBLEMU

2.1 Model choroby zakaźnej

2.1.1 Wykorzystanie R_0

R_0 (Basic reproduction number) to liczba wykorzystywana w epidemiologii do określenia statstycznej liczby zakażeń generowanych przez jedną chorą osobę podczas przebiegu infekcji w populacji praktycznie w całości podatnej na patogen [3]. Przykładowo niekontrolowana infekcja o $R_0 = 2$ w populacji o wielkości n statystycznie wygeneruje $\log_2 n$ przypadków (rys. 2.1)



Rysunek 2.1: Rozprzestrzenianie choroby o różnych wartościach R_0. Źródło: [1].

R_0 jest liczbą zależną w dużej mierze od lokalnego miejsca występowania [4] epidemii-

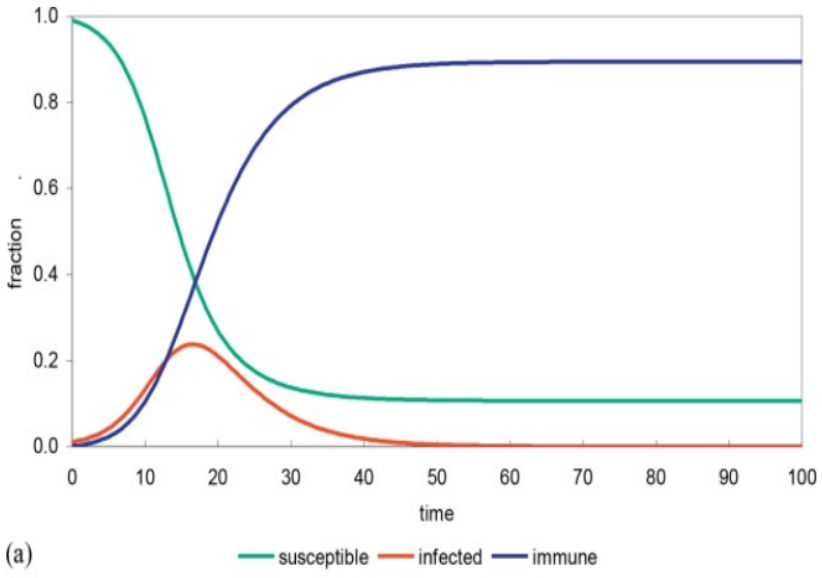
/endemii i jest zazwyczaj obliczana w początkowych fazach jej rozwoju. Ze względu na dostępność tych danych została użyta w naszym modelu do prezentowania rozwoju epidemii.

2.1.2 Wpływ dystansu na prawdopodobieństwo zarażenia

Wpływ odległości na prawdopodobieństwo zakażenia zostało dogłębnie zbadane podczas trwania ostatniej pandemii COVID-19. Badania ([2]) wskazują na wpływ trzymania odległości od innych (potencjalnie zakażonych) osób jako kluczowy element w opanowaniu stanu emidemicznego. Efekt ten będziemy w stanie zaobserwować w wynikach naszych symulacji [7].

2.1.3 Krzywa epidemii

Przeciętny przebieg padnemii zasadniczo odbywa się według określonego schematu - zakładając, że choroba nie jest w 100% śmiertelna, ilość osób chorych jest odwrotnie proporcjonalna do części populacji, która po wyzdrowieniu zyskuje na nią odporność [5] (rys. 2.2). Nasza symulacja przewiduje odwzorowanie tego efektu.



Rysunek 2.2: Stosunek części podatnej na zakażenie, chorej i odpornej na infekcję populacji w czasie. Źródło: [6].

3. Dane porównawcze

Do weryfikacji naszego modelu używamy głównie badań prowadzonych nad niedawną pandemią SARS Covid-19. Chociaż nie dostarczają one specyficznie danych o stosunkowo mikroskopijnym zjawisku zakażania wśród ludzkości (nasza symulacja zakłada najwyżej kilkaset agentów), to wiele źródeł [12] jest bogatych w ogólne, liczbowe dane o chorobie zakaźnej, na którym możemy bazować nasz model, jak i dostarczają przykładów wpływów odległości między osobnikami [13] na skuteczność roznoszenia się choroby.

Te informacje pozwalają nam na uwzględnienie realistycznych czynników wpływających na dynamikę epidemii w naszej symulacji. Bierzemy pod uwagę również jak analiza statystyczna naszej symulacji odnosi się do innych badań [14].

4. MODEL FORMALNY

4.1 Struktura Symulacji

Symulacja opiera się na dwuwymiarowej planszy (E) reprezentującej obszar, na którym poruszają się agenty. Każda komórka planszy może być pusta, zawierać agenta lub być "wyłączona" (np. ściana). Plansza jest zdefiniowana jako równanie (rów. 4.1):

$$E = (X, Y, C) \tag{4.1}$$

gdzie X to szerokość, Y to wysokość, a C to zbiór wszystkich możliwych współrzędnych na planszy.

4.2 Automat komórkowy

Automat komórkowy to pojęcie matematyczne składające się z następujących elementów [11]:

- sieci komórek i przestrzeni D-wymiarowej,
- zbioru s_i stanów pojedynczej komórki, zwykle ten sam dla wszystkich, zawierający k elementów,
- reguła F określająca stan komórki w danym czasie t, zależna od stanu otoczenia tej komórki w tej chwili $s_i(t+1) = F(s_j(t)), j \in O(i)$, gdzie O(i) to otoczenie - rozszerzone sądziedztwo Moora (więcej w podrozdziale 4.4).

4.3 Agent

Agent jest podstawową jednostką w modelu, reprezentującą osobę w populacji pełniącą rolę komórki w automacie komórkowym. Każdy agent posiada unikalne ID, stan zdrowia, położenie na planszy i zdolność do poruszania się. To umożliwia uwzględnienie różnych scenariuszy. Agenty mogą znajdować się w następujących stanach:

1. zdrowy,
2. chory (z objawami infekcji, roznoszący wirusy),
3. zakażony (po kontakcie z wirusem, ale jeszcze nie zainfekowany),
4. wyleczony (odporny).

4.3.1 Poruszanie Agentów

Algorytm poruszania agentów odgrywa kluczową rolę w dynamicznej symulacji rozprzestrzeniania się chorób. Każdy agent ma zdolność do poruszania się po planszy zgodnie z określonymi zasadami. W naszym modelu zakładamy istnienie trzech dostępnych wzorców zachowań.

4.3.1.1 Pseudolosowy Ruch Agentów

Pseudolosowy ruch agentów obejmuje losowy wybór jednego z możliwych kierunków ruchu (północ, południe, wschód, zachód). W każdym kroku symulacji agent może zdecydować się na zmianę położenia według następujących równań (rów. 4.3, 4.4).

Załóżmy, że:

A - bieżące położenie agenta(w danym kroku),
A' - położenie agenta w kolejnym kroku symulacji,
Ax, Ay - położenie agenta na osi poziomej i pionowej planszy,
$A'x, A'y$- położenie agenta na osi poziomej i pionowej planszy w kolejnym kroku,
Dx, Dy - liczba pseudolosowa w dziedzinie liczb całkowitych [-1, 0, 1] losowana w danym kroku,
X - szerokość planszy,

Y - wysokość planszy.
(4.2)

Wówczas:

$$A'x = \begin{cases} X & \text{gdy Ax + Dx > X} \\ 0 & \text{gdy Ax + Dx < 0} \\ Ax + Dx & \text{w przeciwnym wypadku} \end{cases} \tag{4.3}$$

Analogicznie dla A'y:

$$A'y = \begin{cases} Y & \text{gdy Ay + Dy > Y} \\ 0 & \text{gdy Ay + Dy < 0} \\ Ay + Dy & \text{w przeciwnym wypadku} \end{cases} \tag{4.4}$$

4.3.1.2 Zadaniowy ruch agentów

W symulacji zaimplementowany został kontroler bazujący na modelu punktów referencyjnych, mający na celu symulować "zadaniowe" zachowanie ludzi w określonych miejscach - np. w sklepie. Agenty zachowują się według ściśle określonego algorytmu (alg. 1), który prowadzi je od jednego wyznaczonego przez użytkownika zadania (checkpointa) do następnego.

Algorytm (alg. 1) składa się z trzech głównych elementów:

- wybierz cel, do którego będzie poruszał się agent (alg. 2),
- wyznacz najlepszą drogę do niego (alg. 3) za pomocą algorytmu A* (alg. 5),
- symuluj wykonywanie zadania (alg. 4).

Algorithm 1 `Przykład algorytmu poruszania agentów`

Input: Przekazanie mapy terenu i pozycji checkpointów
Input: Przez zadaną ilość iteracji powtarzanie czynności
1: **for** $iteracja$ **in range** $iloscIteracji$ **do**
2: $wybranyCheckpoint \leftarrow$ LosowyCheckpoint($pozycjeCheckpointow$)
3: $najlepszaDroga \leftarrow$ WyznaczNajlepszaDroge($mapaTerenu$, $aktualnaPozycjaAgenta$, $wybranyCheckpoint$)
4: wykonajZadanie($aktualnaPozycjaAgenta$, $wybranyCheckpoint$)
5: **end for**

Algorithm 2 `LosowyCheckpoint`

Input: PozycjeCheckpointow
Output: LosowyCheckpoint
1: $losowyIndex \leftarrow$ LosujLiczbeMiedzy(0, $liczbaCheckpointow - 1$)
2: **return** $pozycjeCheckpointow[losowyIndex]$

Algorithm 3 `WyznaczNajlepszaDroge`

Input: MapaTerenu, Start, Cel
Output: NajlepszaDroga
1: $algorytmAStar \leftarrow$ InicjalizujAlgorytmAStar($mapaTerenu$, $start$, cel)
2: **return** $algorytmAStar.WyznaczNajlepszaDroge()$

Algorithm 4 `wykonajZadanie`

Input: AktualnaPozycjaAgenta, Cel
Output: WykonanieZadania
1: PrzejscieDoStanuUspionego($aktualnaPozycjaAgenta$)
2: WykonajZadaniePrzyCelu(cel)

Aby zapewnić sprawne działanie algorytmu, użyty został powszechnie popularny algorytm znajdywania ścieżki, A* (alg. 5). Algorytm został dostosowany do struktury danych symulacji (zapis ścian, rozmieszczenia agentów).

4.3.1.3 Stadny ruch agentów

Model stada będzie wzorowany na algorytmie stada opracowanym przez Craiga Reynoldsa ([8]). W najbardziej podstawowej wersji, zachowanie agenta w świecie symulacji algorytmu kontrolują trzy zasady:

1. Rozdzielność – sterowanie zapobiegające lokalnym zbiorowiskom.
2. Spójność – sterowanie w kierunku uśrednionego położenia lokalnej grupy.
3. Wyrównywanie – sterowanie w kierunku uśrednionego celu lokalnej grupy.

Ponadto, algorytm modelujący stado (ławicę) zakłada, że w obliczu niebezpieczeństwa osobniki rozproszą się, przedkładając własne życie nad trzymanie się w grupie. W naszej wersji algorytmu zrezygnowaliśmy z takiego zachowania, jako, że tłum zazwyczaj nie ucieka panicznie przed osobnikami przejawiającymi oznaki chorobowe.

Model jest przeznaczony do symulowania bardziej otwartych przestrzeni typu rynek. Agenty poruszają się według algorytmu (alg. 6) łączącego elementy losowości z tendencją do trzymania się w grupie.

Algorithm 5 `Algorytm A*`

Input: Graf reprezentujący mapę terenu G, początkowy wierzchołek $start$, celowy wierzchołek cel
Output: Najkrótsza ścieżka od $start$ do cel

$OtwartaLista \leftarrow$ pusta lista priorytetowa, $ZamknietaLista \leftarrow$ pusty zbiór wierzchołków
Dodaj $start$ do $OtwartaLista$ z kosztem $f(start) = g(start) + h(start, cel)$
Inicjalizuj koszty: $g(start) = 0$, $h(start, cel)$ to heurystyka odległości do celu
while $OtwartaLista$ nie jest pusta **do**
  $aktualny \leftarrow$ wierzchołek o najniższym koszcie f w $OtwartaLista$
  if $aktualny$ jest celem **then**
    return RekonstruujŚcieżkę($start$, cel, $aktualny$)
  end if
  Przenieś $aktualny$ z $OtwartaLista$ do $ZamknietaLista$
  for każdy sąsiad s wierzchołka $aktualny$ **do**
    if s jest w $ZamknietaLista$ **then**
      continue
    end if
    $nowyKoszt \leftarrow g(aktualny) + koszt(aktualny, s)$
    if s nie jest w $OtwartaLista$ lub $nowyKoszt < g(s)$ **then**
      $g(s) \leftarrow nowyKoszt$
      $h(s, cel) \leftarrow$ heurystyka odległości do celu z wierzchołka s
      $f(s) \leftarrow g(s) + h(s, cel)$
      if s nie jest w $OtwartaLista$ **then**
        Dodaj s do $OtwartaLista$ z kosztem $f(s)$
      end if
    end if
  end for
end while
return Brak ścieżki

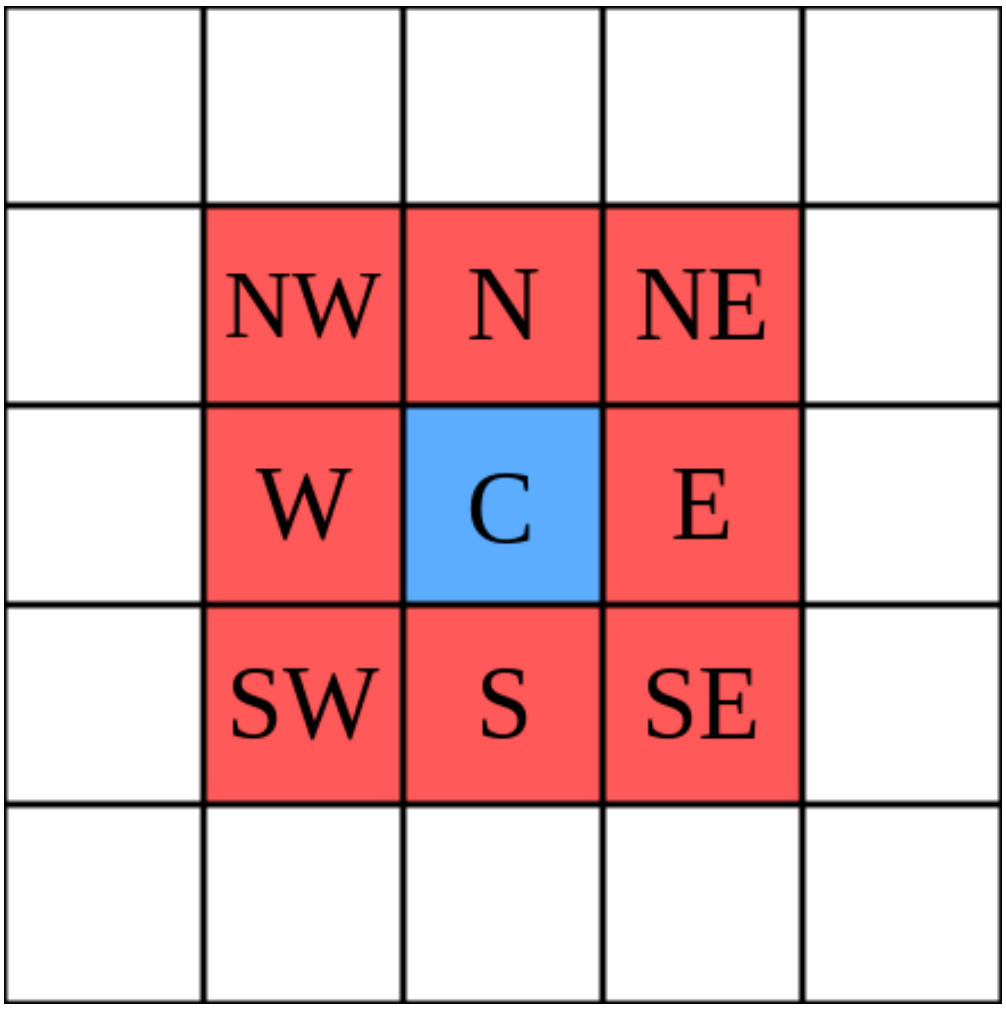
4.4 Implementacja Modelu Automatu Komórkowego z Rozszerzonym Sąsiedztwem Moore'a

Sąsiedztwo Moore'a [9] jest jednym z najczęściej stosowanych sąsiedztw w automatach komórkowych. Każda komórka ma ośmiu sąsiadów, włączając przekątne (rys. 4.1). To oznacza, że każda komórka ma osiem komórek sąsiadujących z nią. Sąsiedztwo Moore'a jest bardziej elastyczne niż sąsiedztwo Von Neumanna i uwzględnia ruch w każdym kierunku.

Algorithm 6 `Algorytm Poruszania się Agenta`

Input: Aktualny stan agenta *stan*
Input: Stała prawdopodobieństwa *staa_prawdopodobienstwa*
Input: Pole widzenia agenta *pole_widzenia*

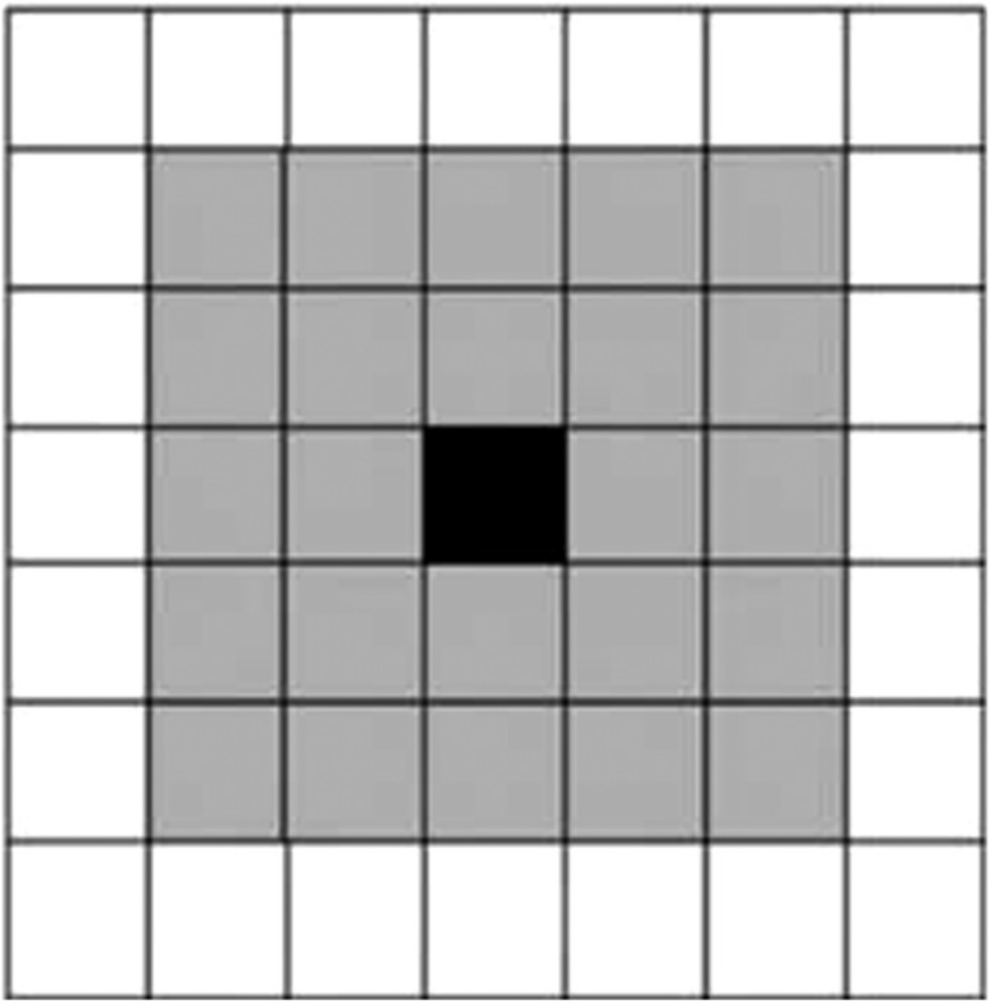
```
1: losowy_stan ← randint()
2: if losowy_stan < staa_prawdopodobienstwa then
3:    go_random() {Przejdź losowo}
4: else
5:    najblizszy_sasiad ← znajdz_najblizszego_sasiada(pole_widzenia)
6:    if najblizszy_sasiad then
7:       zbliz_sie(najblizszy_sasiad) {Zbliż się do najbliższego sąsiada}
8:    else
9:       go_random() {Przejdź losowo, gdy brak sąsiada w polu widzenia}
10:   end if
11: end if
```



Rysunek 4.1: Sąsiedztwo Moore'a [9].

W naszym przypadku ze względu na możliwość roznoszenia się choroby drogą kro-

pelkową na przykład przez kichanie lub oddychanie musimy zastosować rozszerzoną wersję sąsiedztwa Moore'a [10] (rys. 4.2). Będzie ono pokrywać większy obszar niż pierwotne sąsiedztwo.



Rysunek 4.2: Rozszerzone sąsiedztwo Moore'a 5x5 [10].

W naszym modelu, rozszerzone sąsiedztwo Moore'a zostanie wykorzystane do śledzenia ewentualnych kontaktów lub wystarczającego zbliżenia między agentami na siatce. Zakładamy, że najbliższy kontakt agentów to pozycje N, W, E, S zgodnie z rysunkiem (rys. 4.1). Dalsze pozycje innych agentów względem osoby zarażonej (rys. 4.2) będą odpowiadać odpowiednio mniejszemu prawdopodobieństwu zarażenia się chorobą od innego agenta.

4.5 Prawdopodobieństwo Zakażenia

Prawdopodobieństwo zakażenia (P) zależy od odległości między agentami [7]. Zakłada się, że im bliżej chory agent, tym większe prawdopodobieństwo zarażenia. Parametr $min\_distance$ definiuje minimalną odległość, poniżej której prawdopodobieństwo zakażenia rośnie.
Prawdopodobieństwo zakażenia może zostać opisane wzorem (rów. 4.5) dla odległości d osobnika od osoby zakażonej:

$$P_d = \begin{cases} \frac{1}{R_0+d} & \text{dla d} \leqslant min\_distance \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{4.5}$$

5. REALIZACJA PRAKTYCZNA

5.1 Komponenty sprzętowe

Do przeprowadzania symulacji użyte zostały laptopy o następujących procesorach:

- i5 v.Pro,
- i5 8. generacji,
- i7 9. generacji,
- Apple M2.

5.2 Oprogramowanie

Stworzenie projektu opartego na języku Python w wersji 3.10 oraz jego popularnych bibliotekach, takich jak pygame (2.5.2), numpy (1.26.1) i matplotlib (3.7.1) , wynikało z kilku kluczowych motywacji. Po pierwsze, wybór języka Python był podyktowany jego wszechstronnością, czytelnością kodu i ogromną społecznością deweloperską. Python jest jednym z najczęściej używanych języków programowania, co ułatwia współpracę z innymi programistami, a także zapewnia dostęp do szerokiej gamy modułów i narzędzi.

Biblioteka pygame została wykorzystana do obsługi gier, co jest związane z prostotą jej interfejsu oraz szeroką dostępnością funkcji związanych z rysowaniem, obsługą dźwięku i zarządzaniem zdarzeniami. Dzięki temu programowanie gier staje się bardziej intuicyjne, a jednocześnie efektywne.

Z kolei numpy zostało użyte ze względu na jego efektywne operacje na macierzach, co jest niezwykle przydatne w przypadku projektów, które wymagają pracy z danymi numerycznymi. To narzędzie zwiększa wydajność i zoptymalizowane obliczenia, co może być kluczowe w projektach, gdzie manipulacja danymi odgrywa istotną rolę.

Natomiast matplotlib stanowi doskonałe narzędzie do wizualizacji danych. Jego prostota użycia i rozbudowane możliwości pozwalają tworzyć czytelne wykresy i grafiki,

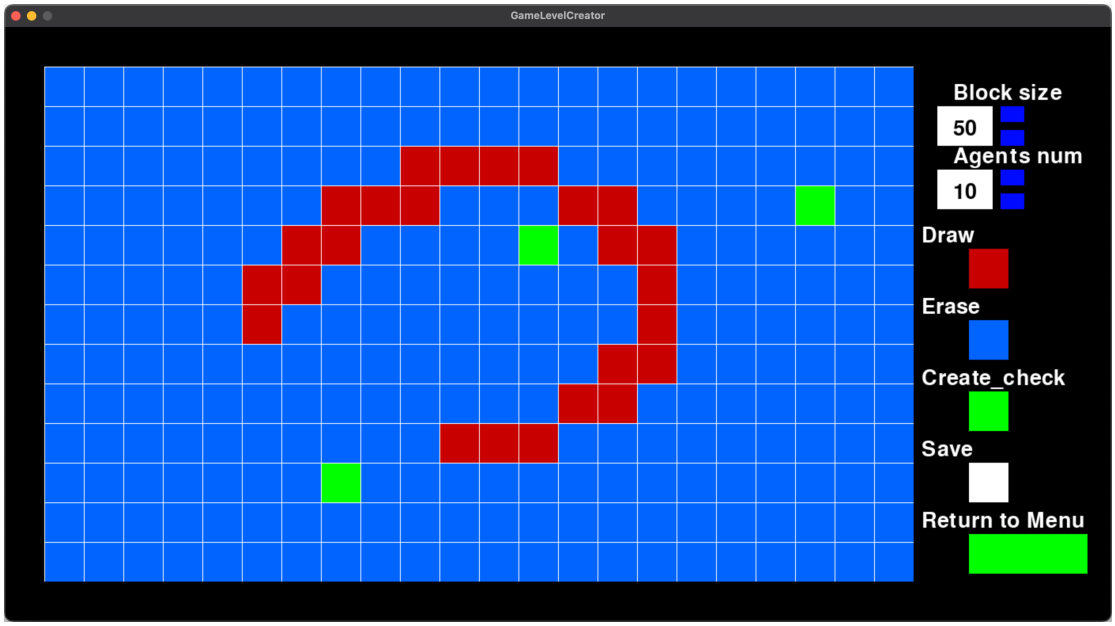
co jest kluczowe w projektach, gdzie istotne jest przedstawienie wyników w formie zrozumiałej dla użytkownika końcowego.

Podsumowując, wybór języka Python oraz konkretnych bibliotek był podyktowany ich popularnością, wszechstronnością, prostotą użycia oraz dostarczeniem efektywnych narzędzi do obsługi konkretnych aspektów projektu, takich jak gry, operacje na macierzach czy wizualizacja danych.

5.3 Szczegóły implementacji

5.3.1 Kreator poziomów

Nasz program udostępnia użytkownikowi możliwość własnoręcznego projetowania poziomów. Użytkownik może ustawić tutaj parametry takie, jak: wielkość pojedynczego bloku (komórki) oraz liczbę agentów. Ma również możliwość rysowania ścian i zaznaczania punktów kontrolnych. Użytkownik ma do swojej dyspozycji prosty system zapisu pracy. Oprawa graficzna kreatora widoczna jest na rysunku (rys. 5.1).



Rysunek 5.1: Kreator poziomów - Level Creator.

5.3.2 Klasa Agent

Klasa `Agent` reprezentuje pojedynczego agenta w symulacji.

Atrybuty

- `id (int)`: Unikalny identyfikator agenta.
- `x (int)`: Współrzędna X pozycji agenta.
- `y (int)`: Współrzędna Y pozycji agenta.
- `position (tuple)`: Krotka reprezentująca współrzędne (x, y) agenta.
- `direction (int)`: Kierunek, w którym porusza się agent.
- `color (tuple)`: Krotka RGB reprezentująca wygląd wizualny agenta.
- `infected (bool)`: Wskazuje, czy agent jest zainfekowany.
- `cured (bool)`: Wskazuje, czy agent został wyleczony.
- `time_infected (int)`: Czas od zainfekowania (w krokach symulacji).
- `controller (Controller)`: Kontroler zachowania agenta.

Metody

- `__init__(id, x, y, direction, color, infected, controller) -> None`: Inicjalizuje nowego agenta.
- `_run_controller()`: Uruchamia kontroler agenta na jednym kroku symulacji.
- `get_cured()`: Oznacza agenta jako wyleczonego, resetując stan zakażenia i zmieniając kolor.

5.3.3 Kontrolery

5.3.3.1 Kontroler ruchu losowego

Atrybuty

- `agent (Agent)`: Referencja do obiektu agenta kontrolowanego przez ten kontroler.
- `width (int)`: Szerokość obszaru symulacyjnego.

- `height (int)`: Wysokość obszaru symulacyjnego.
- `block_size (int)`: Rozmiar bloku lub kroku ruchu agenta.
- `tiles`: Dane reprezentujące kafelki (tiles) w obszarze symulacyjnym.

Metody

- `__init__(width, height, block_size, tiles)`: Inicjalizuje nowy obiekt `IndividualController`.
- `_step()`: losuje kierunek, w którym agent poruszy się w następnym kroku, jeżeli nie natrafił na ścianę.
- `normalise(x, max)`: normalizuje nowe współrzędne agenta do legalnych wartości.

5.3.3.2 Kontroler ruchu stadnego

Atrybuty

- `agents_array (Agent)`: Informacje o innych agentach.
- `agent (Agent)`: Referencja do obiektu agenta kontrolowanego przez ten kontroler.
- `width (int)`: Szerokość obszaru symulacyjnego.
- `height (int)`: Wysokość obszaru symulacyjnego.
- `block_size (int)`: Rozmiar bloku lub kroku ruchu agenta.
- `tiles`: Dane reprezentujące kafelki (tiles) w obszarze symulacyjnym.
- `line_of_sight (int)`: Pole widzenia agenta kontrolowanego przez ten kontroler (w jakim otoczeniu może szukać najbliższego sąsiada).

Metody

- `__init__(width, height, block_size, tiles)`: Inicjalizuje nowy obiekt `IndividualController`.

- `_step()`: losuje, czy agent poruszy się w sposób losowy, czy podąży w stronę najbliższego osbnika.
- `follow_nearest()`: szuka najbliższego sąsiada w polu widzenia. Jeżeli został znaleziony, agent ruszy w jego stronę. Jeżeli nie, agent wykona losowy ruch.
- `_go_random()`: wykonuje losowy ruch.
- `relu()`: Funkcja ta jest używana do generowania wartości reprezentującej odległość w kontekście ruchu agenta.
- `get_distance(self, x, y)`: Ta metoda jest używana do pomiaru odległości między agentem a określonym punktem na obszarze symulacji.
- `normalise(x, max)`: normalizuje nowe współrzędne agenta do legalnych wartości.

5.3.3.3 Kontroler ruchu zadaniowego

Atrybuty

- `agent (Agent)`: Referencja do obiektu agenta kontrolowanego przez ten kontroler.
- `tiles (Tile)`: Dane reprezentujące kafelki w obszarze symulacyjnym.
- `width (int)`: Szerokość obszaru symulacyjnego.
- `height (int)`: Wysokość obszaru symulacyjnego.
- `block_size (int)`: Rozmiar bloku lub kroku ruchu agenta.
- `checkpoints (list)`: Lista punktów kontrolnych, między którymi agent porusza się.
- `waiting (int)`: Licznik oczekiwania na punkcie kontrolnym przed ruchem do następnego.
- `wait_time (int)`: Czas oczekiwania na punkcie kontrolnym przed ruchem do następnego.
- `astar (Astar)`: Obiekt algorytmu A* do wyznaczania ścieżek.
- `path (list)`: Aktualna ścieżka agenta między punktami kontrolnymi.

Metody

- `__init__(width, height, block_size, tiles)`: Inicjalizuje nowy obiekt `IndividualController`.
- `_step()`: Znajduje optymalną drogę do następnego zadania (checkpointa). Jeżeli punkt został osiągnięty, czeka zadaną ilość czasu (wykonuje zadanie)
- `_random_subarray(array, subarray_length)`: zwraca losowy podciąg o zadanej długości

5.3.4 Dokumentacja Klasy InfectionSpread

Klasa `InfectionSpread` reprezentuje mechanizmy rozprzestrzeniania się zakażenia w symulacji.

Atrybuty

- `infected_agent (Agent)`: Referencja do zainfekowanego agenta.
- `block_size (int)`: Rozmiar bloku lub kroku ruchu agenta.
- `R0 (int)`: Współczynnik podstawowej liczby reprodukcji (Basic Reproduction Number).
- `min_distance (float)`: Minimalna odległość wymagana do potencjalnego zakażenia.

Metody

- `__init__(block_size)`: Inicjalizuje nowy obiekt `InfectionSpread`.
- `_get_neighbors(infected_agent, all_agents)`: Zwraca listę sąsiadów zainfekowanego agenta.
- `get_distance(x1, y1, x2, y2)`: Zwraca odległość między dwoma punktami na płaszczyźnie.
- `_reset_neighbors()`: Resetuje listę sąsiadów.
- `distance_x(agent1, agent2)`: Zwraca odległość wzdłuż osi X między dwoma agentami.

- `distance_y(agent1, agent2)`: Zwraca odległość wzdłuż osi Y między dwoma agentami.
- `_spread_infection(potential_infection, num_infected, r0=2, reinfection_probability=0.05)`: Rozprzestrzenia zakażenie wśród potencjalnie zainfekowanych agentów.

5.3.5 Symulacja (gra)

Użytkownik po otwarciu aplikacji może w menu głównym wybrać jedną z przypisanych map do symulacji lub stworzyć nową w komponencie Level creator (rys. 5.1).

Podczas tworzenia mapy definiowane są wielkość siatki oraz liczba agentów. Tworzenie mapy polega na rozmieszczeniu ścian oraz punktów referencyjnych. Punkt referencyjny oznaczają miejsce, w którym agent ma się znaleźć chociaż raz podczas symulacji. Te parametry nie są jednak wymagane ze względu na wybrany kontroler, który determinuje sposób poruszania się agentów.

Wybór kontorlera odbywa się w komponencie Simulation settings, gdzie użytkownik, oprócz niego, wybiera wartość parametru R_0. Następnie po powrocie do menu głównego, użytkownik musi wybrać odpowiednią konfigurację z pliku, którą stworzył w zakładce Level creator (rys. 5.1). Po wybraniu odpowiednich opcji może rozpocząć symulacje klikając przycisk Play.

Podczas symulacji użytkownik ma możliwość przyspieszenia lub zwolnienia jej tempa, co umożliwia dokładniejszą analizę sytuacji w określonym momencie. Można również wygenerować wykresy prezentujące bieżącą liczbę osób chorych, ozdrowieńców oraz zdrowych. Co ważne, aplikacja pozwala na tworzenie kilku wykresów w różnych punktach czasowych, co wspiera szczegółową analizę dynamiki epidemii.

6. REZULTATY SYMULACJI

6.1 Legenda

Do wizualizacji symulacji zostały zastosowane następujące oznaczenia:

- kolor niebieski - wolna przestrzeń, w której mogą poruszać się agenty,
- kolor czerwony - ściany,
- białe kropki - zdrowe agenty,
- czarne kropki - czarne agenty,
- zielone kropki - agenty, które wyzdrowiały.

6.2 Struktura wyników

Wyniki prezentowane będą w następującym schemacie:

- Zrzut z widoku mapy z agentami na początku symulacji,
- Wykres odpowiadający powyższemu,
- Zrzut z widoku mapy z agentami w mniej więcej środku symulacji,
- Wykres do powyższego,
- Zrzut z widoku na końcu symulacji,
- Wykres finałowy.

Powyższe zostanie powtórzone cztery razy, odpowiednio dla każdej symulacji.

6.3 Symulacja losowo poruszającego się tłumu na obszarze odgrodzonym jedną ścianą

6.3.1 Parametry

Do przeprowadzenia symulacji użyte zostały parametry w następującej konfiguracji:

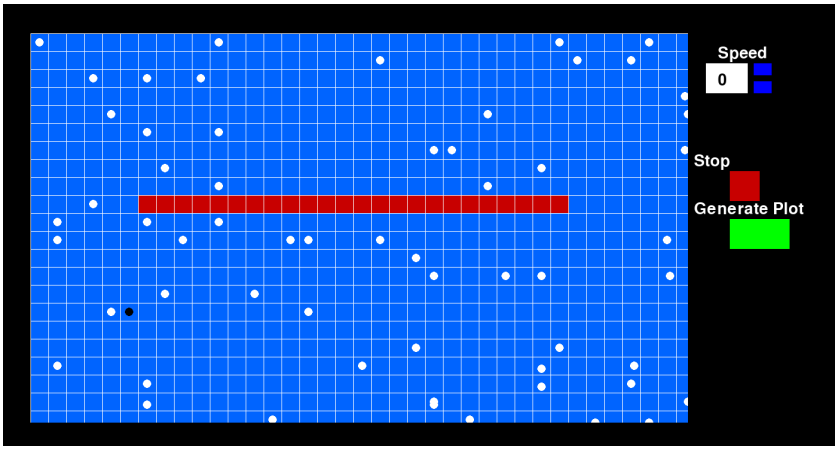
- **liczba agentów**: 60,
- R_0: 2,
- **prawdopodobieństwo ponownego zakażenia osoby ozdrowiałej**: 0,05,
- **sąsiedztwo**: rozszerzone sąsiedztwo Moore'a,
- **kontroler**: losowy.

6.3.2 Przebieg symulacji

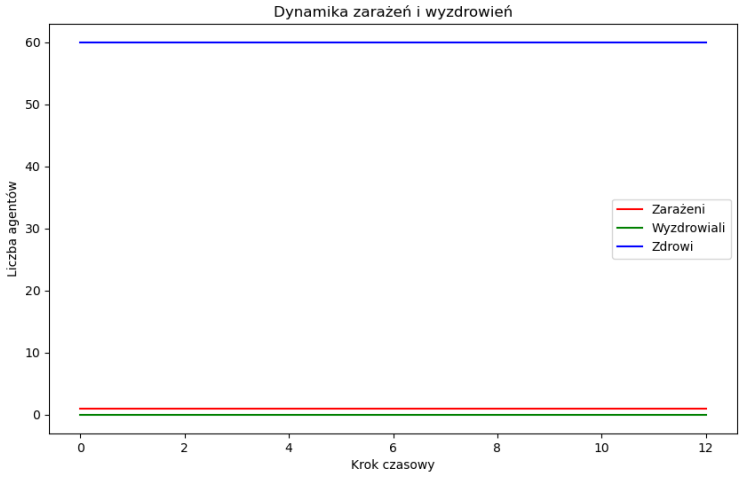
Symulacja obejmuje obszar odgrodzony częściowo przez ścianę (rys. 6.1, 6.3, 6.5). Osobniki poruszają się w sposób losowy.

6.3.2.1 Początek symulacji

W chwili t = 12 Na mapie jest tylko jeden zarażony osobnik (rys. 6.1). Wykresy liczby agentów w danych stanach są całkowicie poziomie (rys. 6.2).



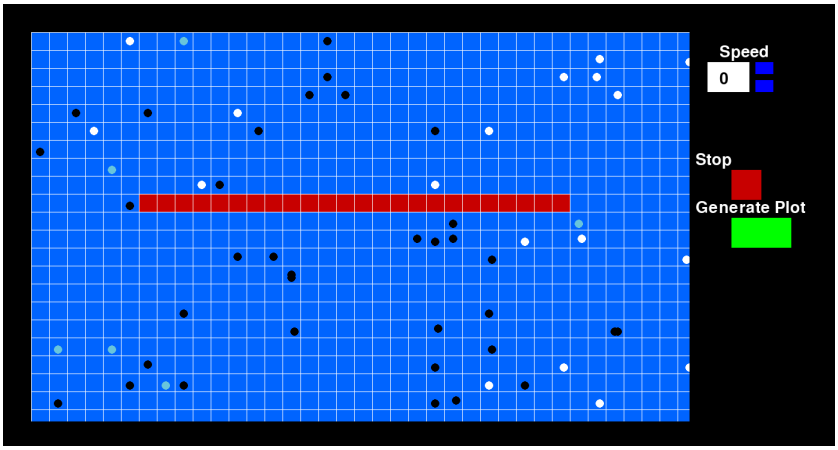
Rysunek 6.1: Stan agentów na mapie na początku symulacji, t = 12.



Rysunek 6.2: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 12.

6.3.2.2 Środek symulacji

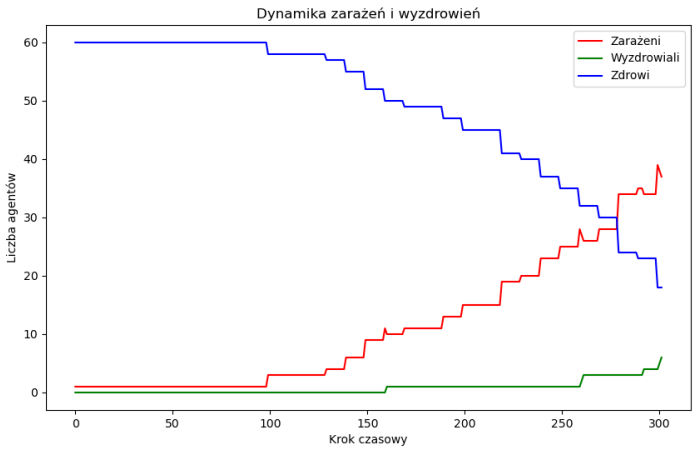
W środku symulacji(t = 300) większość osobników jest już zarażona, pojawiają się też pierwsi ozdrowieńcy(rys. 6.3). Na wykresie(rys. 6.4) Widać liniowy spadek liczby osób zdrowych przy jednoczesnym liniowym wzroście liczby osób chorych.



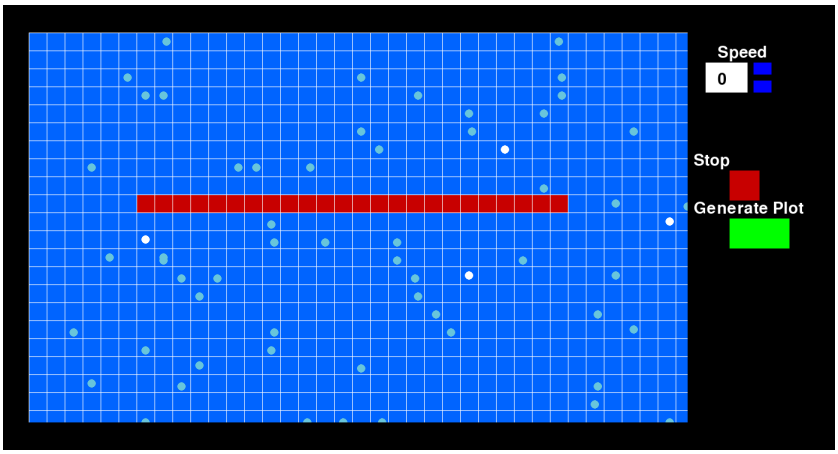
Rysunek 6.3: Stan agentów na mapie w środku symulacji, t = 300.

6.3.2.3 Koniec symulacji

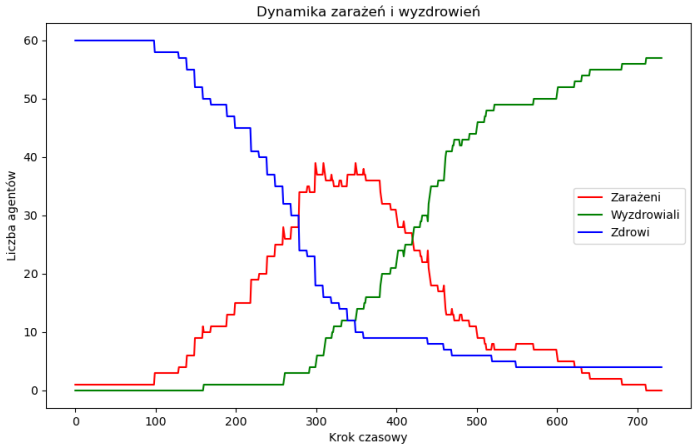
Na rysunku (rys. 6.5) widać że nie ma już żadnej chorej osoby, są tylko 3 osobniki zdrowe i ozdrowieńcy. Sytuacja pokrywa się z wykresem na rysunku 6.6, gdzie po wyraźnym piku liczba osób chorych spadła do zera.



Rysunek 6.4: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 300.



Rysunek 6.5: Stan agentów na mapie na końcu symulacji, t = 750.



Rysunek 6.6: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 750.

6.4 Symulacja konferencji

6.4.1 Parametry

Do przeprowadzenia symulacji użyte zostały parametry w następującej konfiguracji:

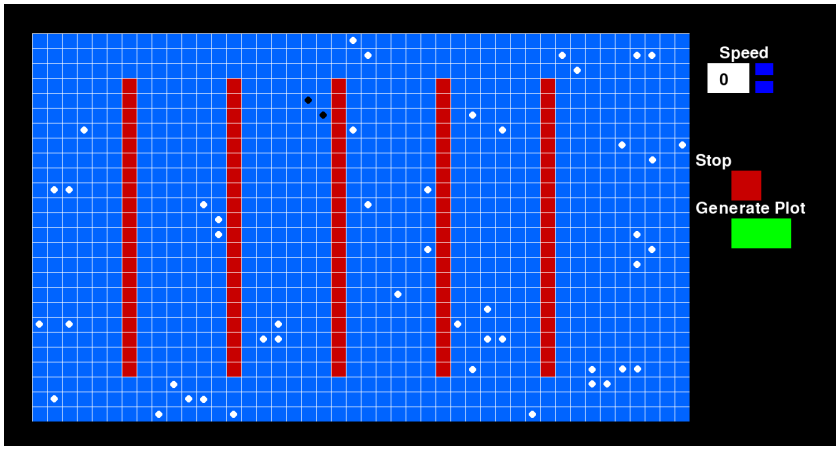
- **liczba agentów**: 50,
- R_0: 2,
- **prawdopodobieństwo ponownego zakażenia osoby ozdrowiałej**: 0,05,
- **sąsiedztwo**: rozszerzone sąsiedztwo Moore'a,
- **kontroler**: stadny.

6.4.2 Przebieg symulacji

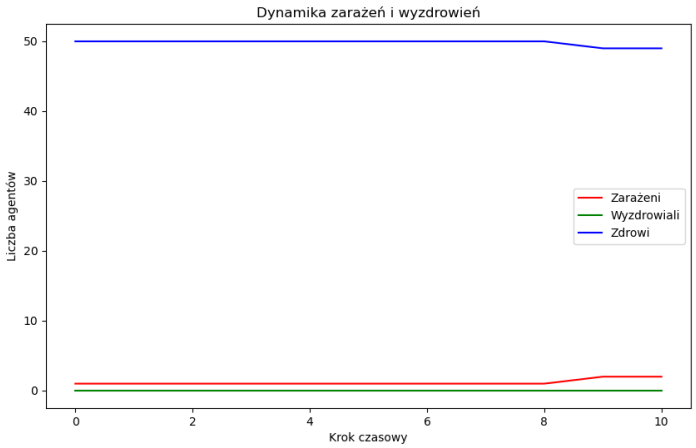
Symulacja obejmuje obszar upodobniony do sali konferencyjnej (ściany zastępują rzędy krzeseł (rys. 6.7, 6.9, 6.11). Osobniki mają tendencję do zbierania się w trzymające dystans grupy. Dane statystyczne symulacji zostały zebrane w postaci wykresów (rys. 6.8, 6.10, 6.11).

6.4.2.1 Początek symulacji

Podobnie jak w poprzednim przypadku, wykresy są niemal poziome, z tą różnicą że dla chwili t = 10 zdążył się zarazić już jeden agent(rys. 6.7, 6.8).



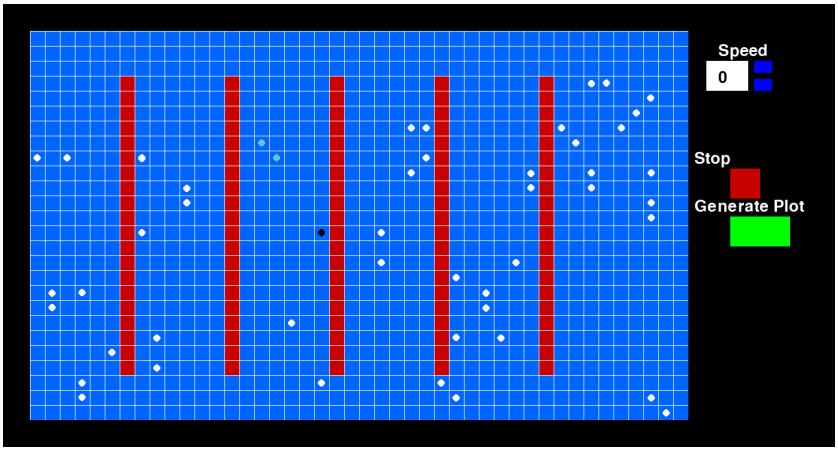
Rysunek 6.7: Stan agentów na mapie na początku symulacji, t=10



Rysunek 6.8: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie t = 10.

6.4.2.2 Środek symulacji

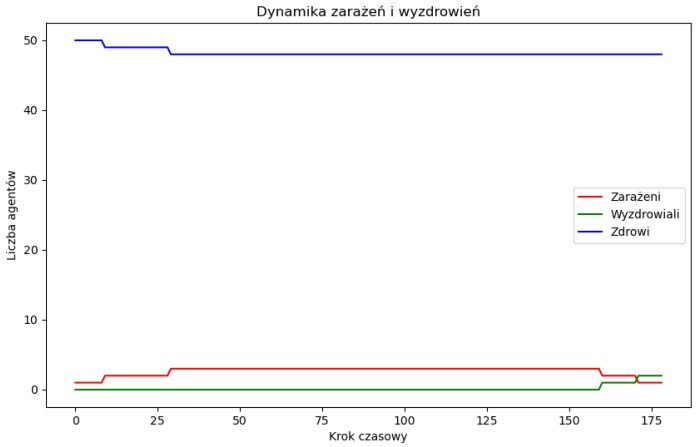
Jak widać na rys. 6.9, ze względu na trzymanie dużych dystansów między grupami, choroba nie miała okazji rozprzestrzenić się na większą ilość osobników. Wykres 6.10 pokazuje, nie doszło do epidemii.



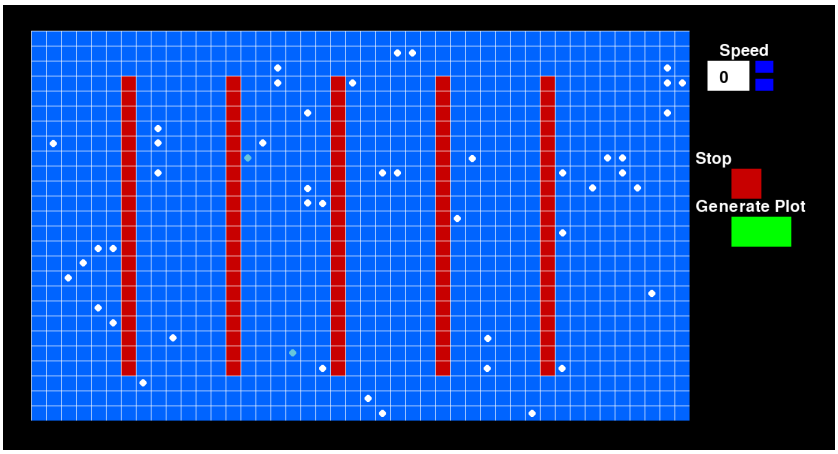
Rysunek 6.9: Stan agentów na mapie w momencie t = 175 symulacji.

6.4.2.3 Koniec symulacji

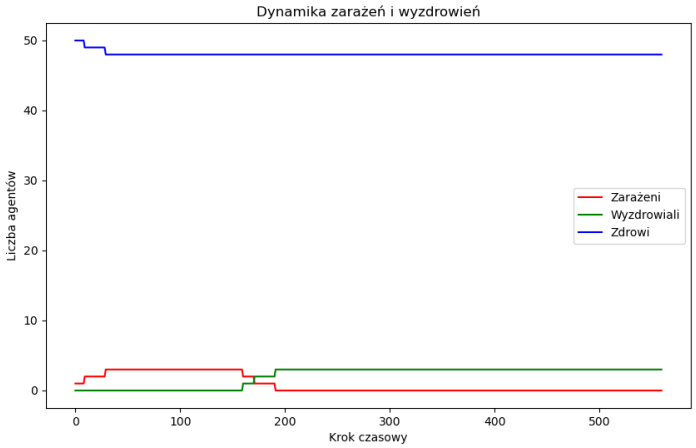
Na mapie (rys. 6.11) brak osobników zakażonych. Epidemia nie była w stanie się rozprzestrzenić.



Rysunek 6.10: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie t = 175.



Rysunek 6.11: Stan agentów na mapie w momencie t = 600 symulacji.



Rysunek 6.12: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie t = 600.

6.5 Symulacja supermarketu

6.5.1 Parametry

Do przeprowadzenia symulacji użyte zostały parametry w następującej konfiguracji:

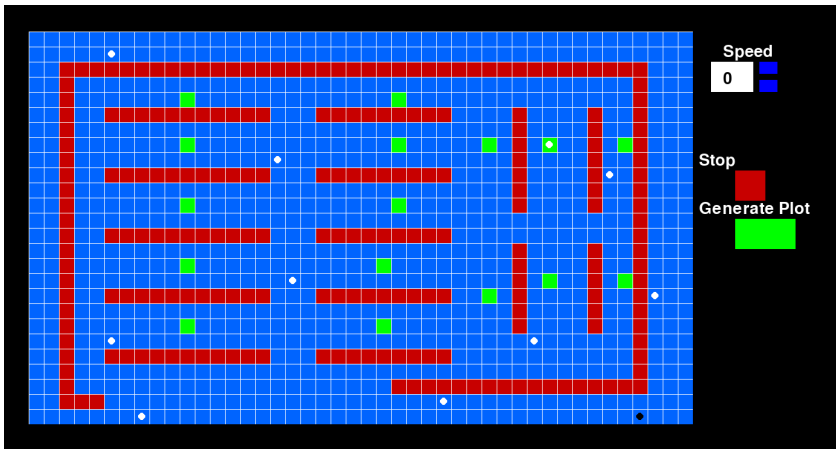
- **liczba agentów**: 10,
- R_0: 2,
- **prawdopodobieństwo ponownego zakażenia osoby ozdrowiałej**: 0,05,
- **sąsiedztwo**: rozszerzone sąsiedztwo Moore'a,
- **kontroler**: zadaniowy.

6.5.2 Przebieg symulacji

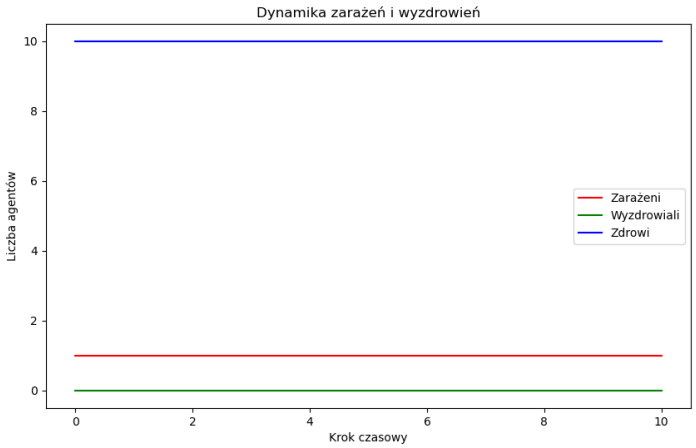
W tej symulacji podjęta została próba odwzorowania supermarketu(patrz rys. 6.13, 6.15, 6.17). Ściany w tym przypadku odwzorowują półki sklepowe. Agenty odwiedzają tzw. punkty referencyjne, czyli wyszczególnione komórki. Co ważne, punkty referencyjne są dla wszystkich agentów takie same, co istotnie wpływa na rozwój epidemii.

6.5.2.1 Początek symulacji

W chwili t = 10 na mapie jest tylko jeden zarażony osobnik (rys. 6.13). Wykresy liczby agentów w danych stanach są całkowicie poziomie (rys. 6.14).



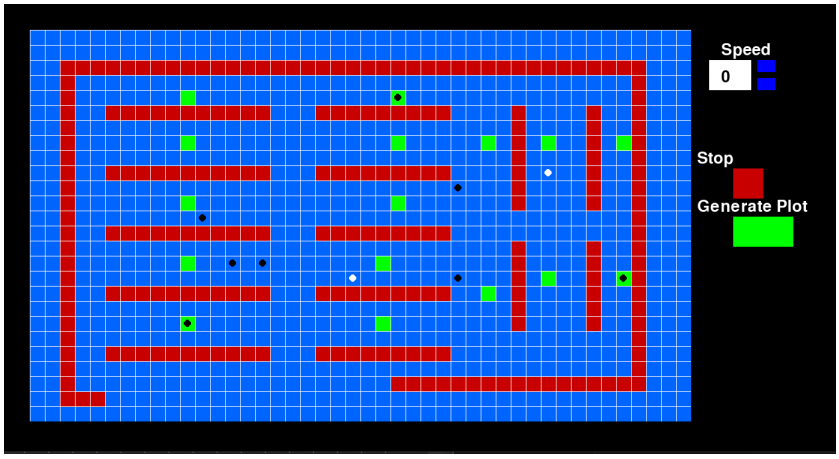
Rysunek 6.13: Stan agentów na mapie na początku symulacji, t = 10.



Rysunek 6.14: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 10.

6.5.2.2 Środek symulacji

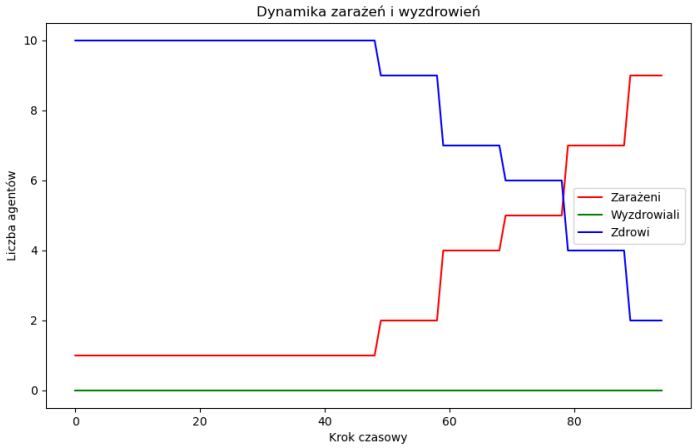
W środku symulacji(t = 95) większość osobników jest już zarażona(rys. 6.15). Na wykresie(rys. 6.16) Widać szybki liniowy spadek liczby osób zdrowych przy jednoczesnym liniowym wzroście liczby osób chorych. Generalnie widać o ile szybciej choroba roznosi się gdy agenty chodzą w te same miejsca.



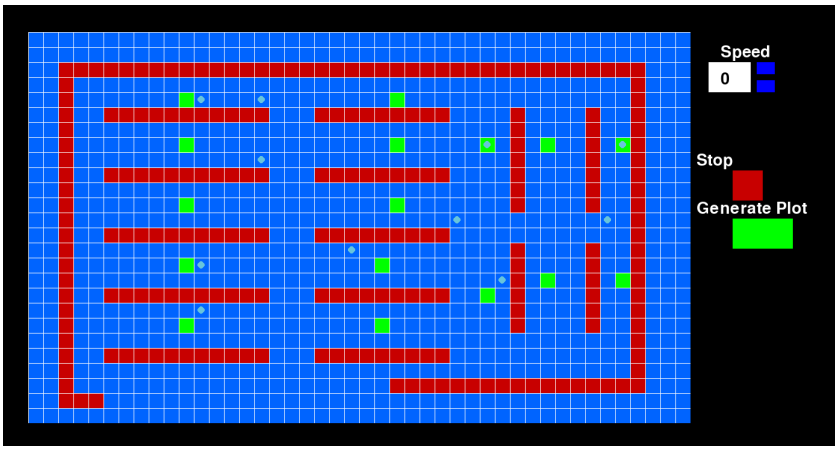
Rysunek 6.15: Stan agentów na mapie w środku symulacji, t = 95.

6.5.2.3 Koniec symulacji

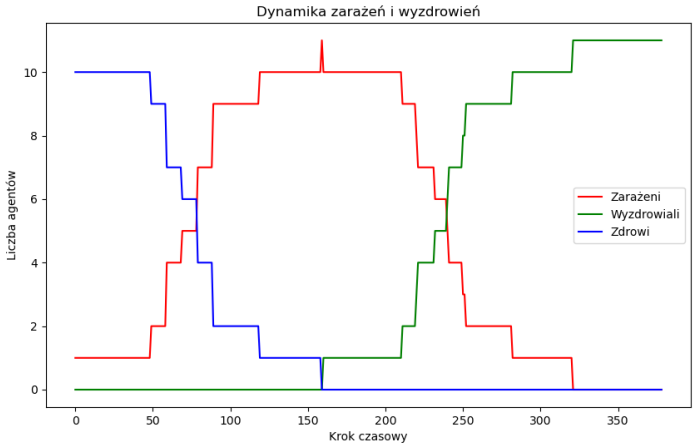
Na rysunku (rys. 6.17) widać że nie ma już żadnej chorej ani zdrowejosoby, są tylko ozdrowieńcy. Na wykresie(rys. 6.18)widać jak szybko epidemia przebiegła - już w t = 150 nie było żadnych osób zdrowych, a w t = 320 nie było już żadnej chorej osoby).



Rysunek 6.16: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 95.



Rysunek 6.17: Stan agentów na mapie na końcu symulacji, t = 390.



Rysunek 6.18: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 390.

6.6 Symulacja dużej liczby agentów na dużej przestrzeni

6.6.1 Parametry

Do przeprowadzenia symulacji użyte zostały parametry w następującej konfiguracji:

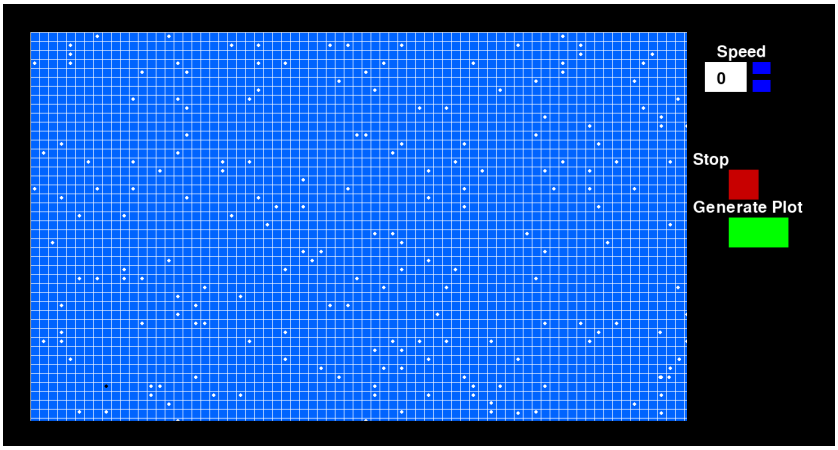
- **liczba agentów**: 200,
- R_0: 2,
- **prawdopodobieństwo ponownego zakażenia osoby ozdrowiałej**: 0,05,
- **sąsiedztwo**: rozszerzone sąsiedztwo Moore'a,
- **kontroler**: losowy.

6.6.2 Przebieg symulacji

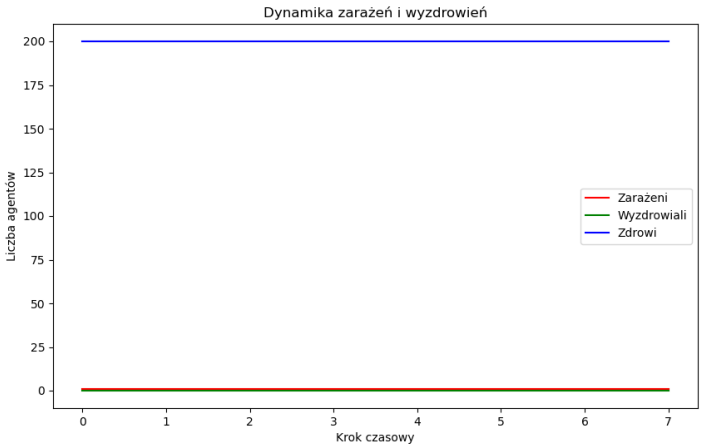
Symulacja obejmuje hipotetyczne sytuacje dużej liczby osób na otwartej przestrzeni bez ścian, którzy poruszają się w sposób losowy (rys. 6.19, 6.21, 6.23).

6.6.2.1 Początek symulacji

W chwili t = 7 Na mapie jest tylko jeden zarażony osobnik (rys. 6.13). Wykresy liczby agentów w danych stanach są całkowicie poziomie (rys. 6.14).



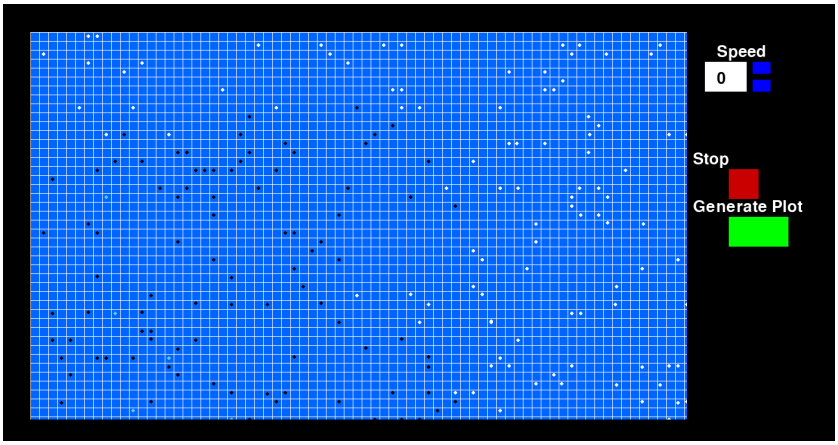
Rysunek 6.19: Stan agentów na mapie na początku symulacji, t = 7.



Rysunek 6.20: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 7.

6.6.2.2 Środek symulacji

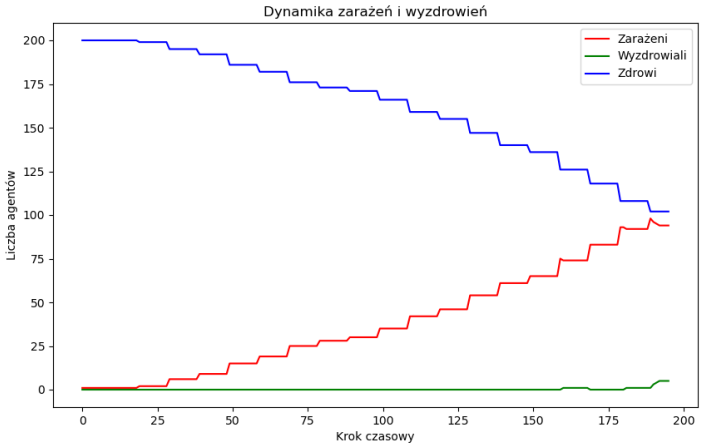
W środku symulacji(t = 200) około połowa osobników jest już zarażona(rys. 6.21). Na wykresie(rys. 6.22) Widać szybki liniowy spadek liczby osób zdrowych przy jednoczesnym liniowym wzroście liczby osób chorych.



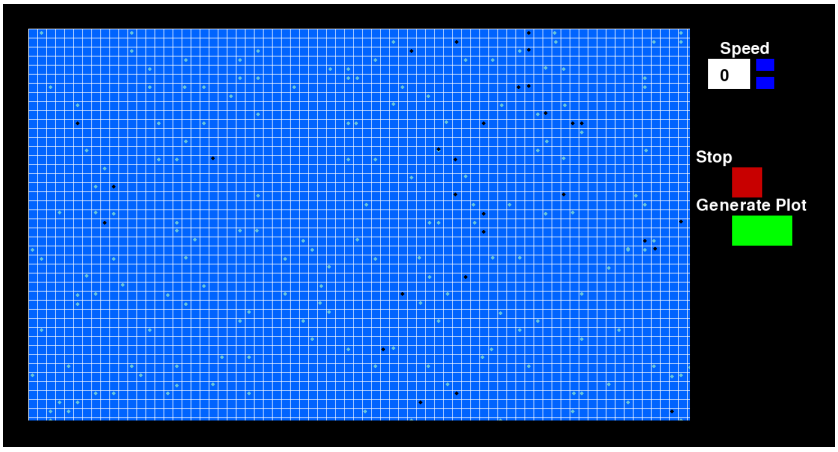
Rysunek 6.21: Stan agentów na mapie w środku symulacji, t = 200.

6.6.2.3 Koniec symulacji

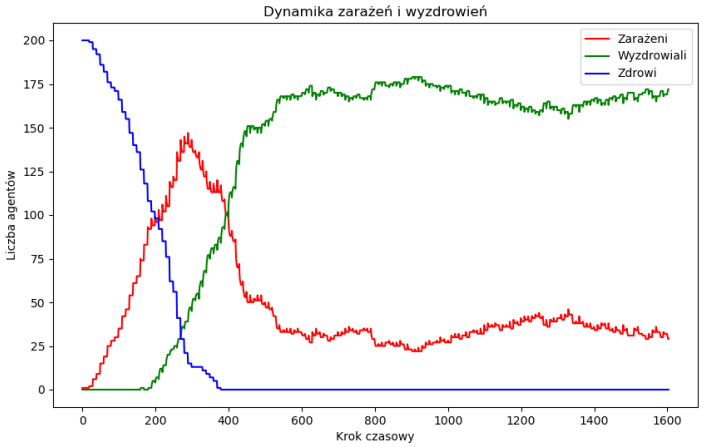
Na rysunku (rys. 6.23) nie ma już żadnej zdrowej osoby. Generalnie wykres(rys. 6.24 jest podobny do rys. 6.6, z tą różnicą, że widać ponowne zakażenie się ozdrowieńców(ok. t = 1300).



Rysunek 6.22: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 200.



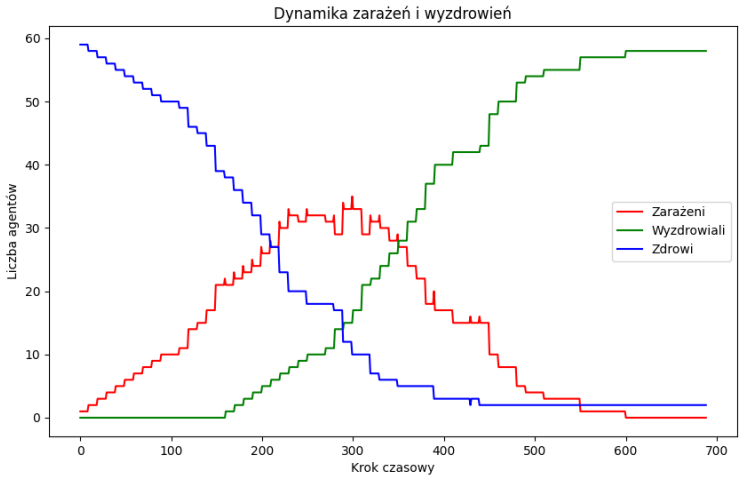
Rysunek 6.23: Stan agentów na mapie na końcu symulacji, t = 1600.



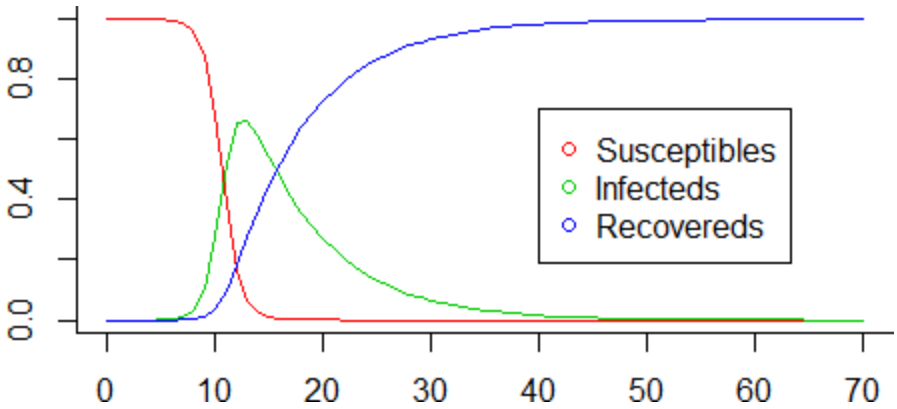
Rysunek 6.24: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie, t = 1600.

7. DYSKUSJA WYNIKÓW

Najbardziej zadowalającym wynikiem naszej symulacji jest fakt, że udało nam się zasymulować rzeczywistą zależność między osobnikami chorymi, zdrowymi i odpornymi w czasie. Obrazowanie danych z naszej symulacji (rys. 7.1) wykazuje te same zależności co inne modele badające dynamikę infekcji ([15]) (rys. 7.2).

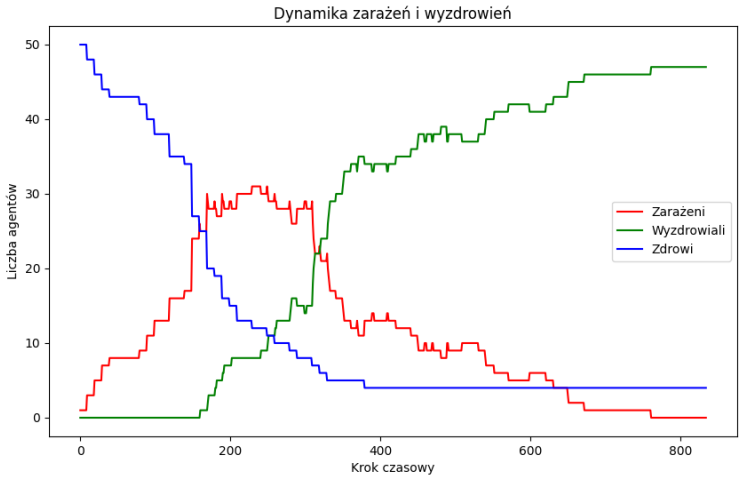


Rysunek 7.1: Wykresy pokazujące zależność liczby chorych, zdrowych i wyzdrowiałych osobników w czasie w symulacji z użyciem kontrolera losowego.

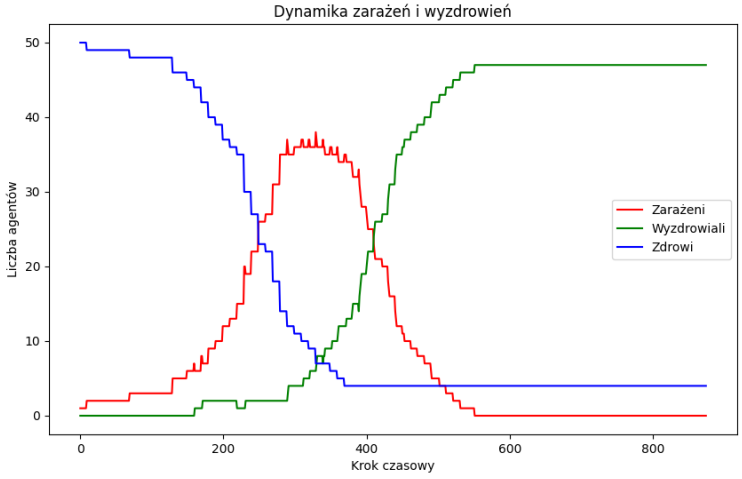


Rysunek 7.2: Model Susceptible-Infected-Recovered (SIR) bazowany na badaniach rzeczywistych epidemii. Źródło: [15].

Oczywiście, na przebieg każdej symulacji duży wpływ ma losowość. Ze względu na m.in. losowe położenie początkowe agentów symulacja może przedstawiać skrajnie różne przebiegi epidemii. Statystyki (rysunki 7.3, 7.4, 7.5) przebiegu epidemii na tej samej mapie (symulacja konferencji, rys. 6.13) różnią się, przy czym symulacja druga wykazała najbardziej gwałtowny przebieg epidemii, natomiast trzecia najłagodniejszy.

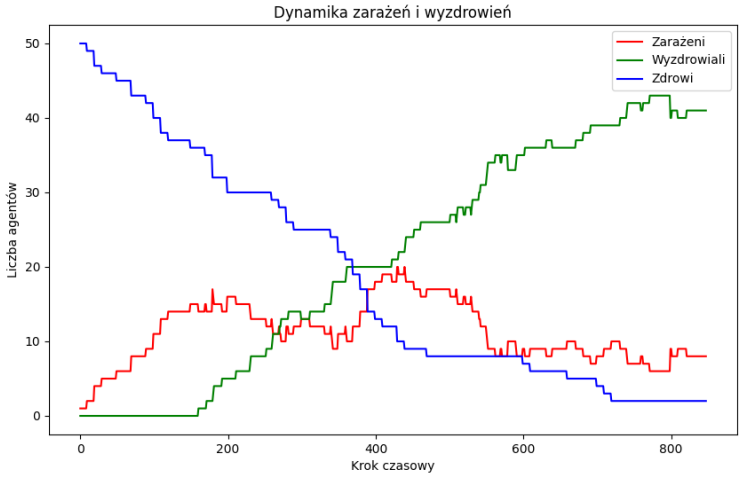


Rysunek 7.3: Przebieg symulacji w czasie ok. 1 min. podczas pierwszej próby.

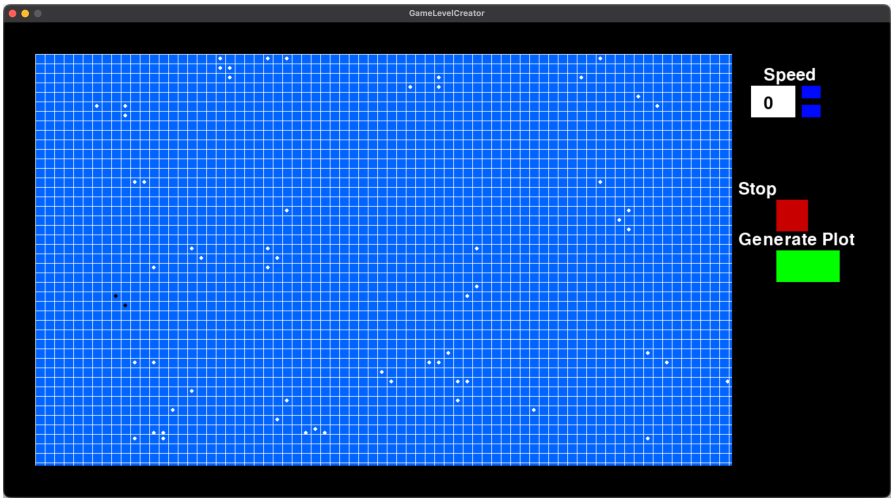


Rysunek 7.4: Przebieg symulacji w czasie ok. 1 min. podczas drugiej próby.

Ciekawym zjawiskiem, które można było zaobserwować w naszej symulacji, szczególnie przy użyciu modelu stadnego był wpływ tzw. *social distancing*, czyli zapobiegania infekcji przez trzymanie odległości od innych ludzi. Chore osobniki nie były w stanie zarażać trzymających się z daleka grup, co skutkowało szybkim wygaśnięciem epidemii (rys. 7.6).



Rysunek 7.5: Przebieg symulacji w czasie ok. 1 min. podczas trzeciej próby.



Rysunek 7.6: Epidemia zatrzymana na niewielkiej liczbie osobników.

8. PODSUMOWANIE

Największym polem do rozwoju pozostaje oprawa graficzna. Wybrana technologia (pygame), chociaż broniła się łatwością implementacji, oferuje bardzo prymitywną grafikę. Strona wizualna programu nie jest przez to zbyt estetyczna oraz nie pracuje zbyt płynnie. Na potrzeby projektu GUI spełnia swoje zadanie, jednak w wypadku chęci kontynuowania produktu na pewno pierwszym punktem do rozważenia byłoby zmienienie technologii.

Przy ponownej implementacji programu na pewno możnaby pomyśleć nad lepszą optymalizacją oraz czystością kodu. Oczywiście, obecna implementacja została w miarę możliwości zoptymalizowana i uporządkowana, jednak miejscami widać piętna projektu ewolującego wraz z rozwijająca się z czasem koncepcją.

Z całą pewnością można jednak stwierdzić, że symulacja spełniła podstawowe założenia projektu - pozwala na obserwację znanych z prawdziwego życia mechanik rozprzestrzeniania chorób zakaźnych. Zastosowane różnorodne kontrolery oraz element przypadku pozwalają na przeprowadzanie symulacji o bardzo interesujących rezultatach. Główne cele projektowe zostały osiągnięte.

Bibliografia

[1] https://en.wikipedia.org/wiki/Basic_reproduction_number

[2] Qian M, Jiang J. COVID-19 and social distancing. Z Gesundh Wiss. 2022;30(1):259-261. doi: 10.1007/s10389-020-01321-z. Epub 2020 May 25. PMID: 32837835; PMCID: PMC7247774. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7247774/

[3] P. Delamater, E. Street, T. Leslie, Y. Yang, K. Jacobse, *Complexity of the Basic Reproduction Number (R_0)*, https://wwwnc.cdc.gov/eid/article/25/1/17-1901_article

[4] F. Guerra, S. Bolotin, G. Lim, J. Heffernan, S. Deeks, Ye Li, N. Crowcroft, *The basic reproduction number (R_0) of measles: a systematic review*, http://materias.df.uba.ar/fca2021c1/files/2019/02/The_basic_reproduction_number.pdf

[5] J. Ma, *Estimating epidemic exponential growth rate and basic reproduction number*, https://www.sciencedirect.com/science/article/pii/S2468042719300491

[6] E. Karkri, M. Benmir, *Some key concepts of mathematical epidemiology Jaafar*, https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/basic-reproduction-number

[7] T. Daghriri, O. Ozmen, *Quantifying the Effects of Social Distancing on the Spread of COVID-19*, https://pubmed.ncbi.nlm.nih.gov/34071047/

[8] https://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/

[9] *Moore neighborhood - Wikipedia*, [Accessed: November 21, 2023], https://en.wikipedia.org/wiki/Moore_neighborhood

[10] S. Djemame, *Training Cellular Automata with Extended Neighborhood for Edge Detection*, https://ceur-ws.org/Vol-2904/14.pdf

[11] K. Kułakowski, *Automaty komórkowe* http://www.ftj.agh.edu.pl/~kulakowski/AC/skrypt.pdf

[12] https://covid.cdc.gov/covid-data-tracker/#maps_new-admissions-rate-county

[13] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9181815/

[14] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6926909/

[15] Khatar, Zakaria and Bouattane, Omar and Bentaleb, Dounia, 09.2019, strony 4579-4588, COMPLEX NETWORKS BASED ALGORITHM FOR INFECTIOUS DISEASES SPREAD MODELING, vol.97, Journal of Theoretical and Applied Information Technology