Politechnika
Śląska

PROJEKT INŻYNIERSKI

Aplikacja do symulacji rozprzestrzeniania się zarażeń

Jakub CIOŁEK
Nr albumu: 295618

Kierunek: Infromatyka
Specjalność: Bazy danych i Inżynieria Systemów

PROWADZĄCY PRACĘ
Dr inż. Ewa Płuciennik
KATEDRA Informatyki Stosowanej
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2023

Tytuł pracy

Aplikacja do symulacji rozprzestrzeniania się zarażeń

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczone

Modelowanie epidemii, Aplikacja edukacyjna, Bezpieczeństwo zdrowotne, Interaktywne narzędzie edukacyjne, Matematyczne modele epidemiologiczne

Thesis title

Application for simulating the spread of infections.

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Keywords

Epidemic modeling, Educational application, Health safety, Interactive educational tool, Mathematical epidemiological models

Spis treści

Rozdział 1

Wstęp

W świecie, który dopiero co doświadczył globalnej pandemii COVID-19, zauważamy potrzebę skutecznych narzędzi zarówno do przewidywania rozprzestrzeniania się infekcji, jak i podnoszenia świadomości społeczeństwa na temat konieczności przestrzegania restrykcji i ochrony zdrowia. Pandemia wywołała potrzebę innowacyjnych rozwiązań, w obszarze przewidywania i zrozumienia dynamiki rozprzestrzeniania się wirusa. W kontekście informatyki, praca skupia się na wykorzystaniu komputerów do opracowania aplikacji, która nie tylko pozwala na modelowanie dynamicznych scenariuszy rozprzestrzeniania się wirusa, ale także stawia na edukację społeczną w zakresie efektywnych praktyk prewencyjnych.

Dziedzina informatyki w znacznym stopniu przyczynia się do rozwiązania problemów związanych z pandemią. Mając do dyspozycji technologię, jesteśmy w stanie opracować zaawansowane algorytmy symulacyjne, które umożliwiają modelowanie złożonych interakcji społecznych i ruchu ludzi w różnych środowiskach. Komputery stają się potężnym narzędziem do analizy danych, identyfikowania wzorców i prognozowania potencjalnych scenariuszy rozprzestrzeniania się infekcji.

Symulacje komputerowe pozwalają nam przewidywać, jak różne warunki środowiskowe i społeczne wpływają na tempo i zasięg rozprzestrzeniania się wirusa. Ponadto, algorytmy sztucznej inteligencji mogą być używane do analizy zachowań społecznych, co pozwala na lepsze zrozumienie, jak ludzie reagują na różne sytuacje i jakie czynniki wpływają na przestrzeganie restrykcji.

Nasza praca w obszarze informatyki nie tylko skupia się na technicznej strukturze aplikacji, ale również na zastosowaniu narzędzi informatycznych w celu zwiększenia świadomości społecznej. Komputery służą jako platforma, na której możemy nie tylko symulować scenariusze, ale także efektywnie komunikować się z użytkownikami spoza środowiska medycznego, edukując ich na temat istoty zachowania się w sposób, który zmniejsza ryzyko zakażenia.

W kolejnych rozdziałach przedstawione zostaną dokładne metody i technologie, jakie wykorzystano do implementacji aplikacji, oraz skoncentrujemy się na roli informatyki w

rozwiązaniu współczesnych wyzwań zdrowotnych.

Rozdział drugi skupiony jest na przeglądzie istniejących modeli symulacji zarażeń, opartym na dogłębnej analizie dostępnej literatury. Ma na celu zidentyfikować różne podejścia i metody, które zostały wykorzystane w modelowaniu rozprzestrzeniania się infekcji.

Rozdział trzeci posłuży do przedstawienia wymagań projektowych i narzędzi, które posłużą do ich realizacji. Analiza potrzeb funkcjonalnych i technicznych pozwoli na wybór odpowiednich technologii i narzędzi programistycznych

W kolejnych dwóch rozdziałach przedstawiono odpowiednio specyfikacja zewnętrzna i wewnętrzna aplikacji. Zdefiniowany zostanie interfejs użytkownika, funkcjonalności dostępne dla użytkowników końcowych oraz scenariusze użycia. Następnie opis architektury, struktury kodu i wszystkich kluczowych elementów wewnętrznych.

Szósty rozdział poświęcony został procesom weryfikacji i walidacji stworzonej aplikacji. Opisuje wykorzystane scenariusze testowe.

Ostatni rozdział to podsumowanie całej pracy, uwzględniające wnioski powstałe z realizacji projektu oraz ewentualne kierunki dalszych rozwojów.

Rozdział 2

Modele symulacji rozprzestrzeniania się zarażeń

Aby skutecznie symulować rozprzestrzenianie się zarażeń, konieczne jest w pierwszej kolejności zrozumienie mechanizmów, które kierują postępującą zarazą. Początek naszej pracy powinien poprzedzić dogłębne zbadanie natury patogenu, jego zdolności i ograniczeń wynikających z procesów selekcji naturalnej. Wirus, aby przetrwać, musi zdolnością zarażania przewyższać zdolność zabijania, co sprowadza się do utrzymania współczynnika rozprzestrzeniania większego niż 1. Dodatkowo, uwzględnienie okresu inkubacji jest kluczowe, ponieważ wirus potrzebuje czasu na rozmnożenie się w organizmie nosiciela.

Jednakże, natura patogenu to tylko jeden z elementów, na które należy zwrócić uwagę w kontekście symulacji. Równie istotnym aspektem jest człowiek jako ofiara. Analiza funkcjonowania współczesnego społeczeństwa pomoże nam określić skalę, na jaką może rozprzestrzeniać się zaraza. Zrozumienie tego kontekstu umożliwi nam lepsze odzwierciedlenie rzeczywistości w modelowaniu.

Zebraną wiedzę należy następnie przełożyć na język matematyki i modelować ją komputerowo. W tym procesie istotne jest zidentyfikowanie obszarów, które mogą być uproszczone, oraz tych, które wymagają szczegółowego odwzorowania, aby osiągnąć postawione cele symulacji. W ten sposób, połączenie wiedzy o patogenie i społeczeństwie, przełożone na modele matematyczne, pozwoli nam skutecznie symulować i analizować procesy rozprzestrzeniania się zarażeń.

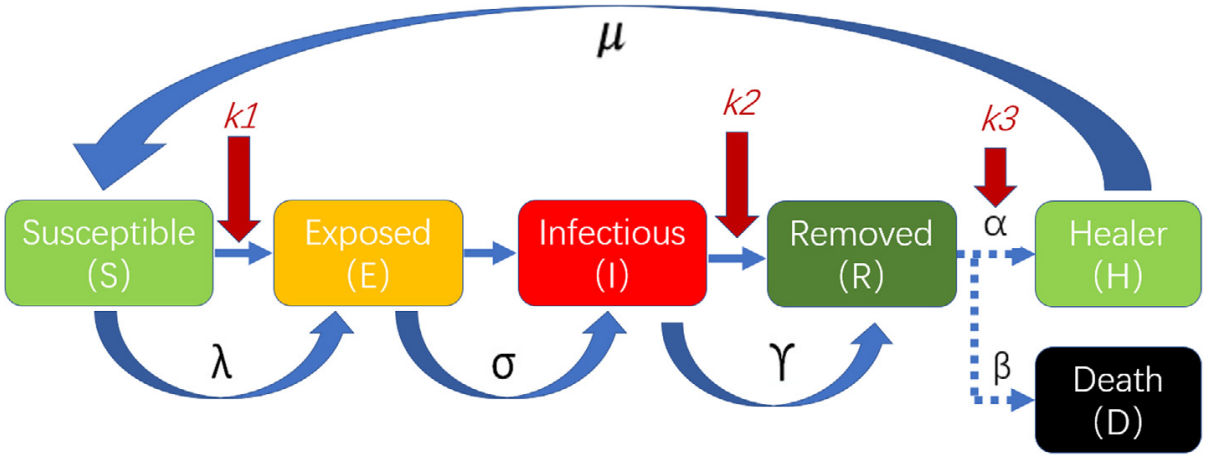
Mimo wcześniejszych prób i starań badaczy nad zjawiskiem rozprzestrzeniania się patogenów, znaczące postępy i wzmożone zainteresowanie tematem pojawiły się dopiero w latach 20. ubiegłego wieku. Świat po I wojnie światowej stanął przed pandemią grypy hiszpanki, która zarażając 1/3 ówczesnej populacji i powodując więcej ofiar niż dopiero co zakończony globalny konflikt zbrojny, spowodowała pilną potrzebę zrozumienia i kontrolowania takich masowych zjawisk. W okresie tym, w odpowiedzi na potrzebę zrozumienia dynamiki pandemii grypy hiszpanki, powstał jeden z pierwszych matematycznych modeli symulacyjnych dotyczących rozprzestrzeniania się chorób zakaźnych, znany jako model

SIR (podatni-zainfekowani-ozdrowieńcy). Model ten, opracowany w tamtych latach, stał się punktem wyjścia dla wielu kolejnych prac nad matematycznym modelowaniem epidemii, ukazując potencjał tego podejścia do zrozumienia i przewidywania rozprzestrzeniania się patogenów w społeczeństwie.

2.1 Modele bazujące na SIR

We współczesnych badaniach często rozwija się model SIR tak aby mógł lepiej dokładniej odzwierciedlać rozprzestrzenianie się choroby. Takimi modyfikacjami najczęściej są dalsze podzielenie populacji na grupy czy dodanie dodatkowych czynników wpływających na zarazę. Jednym z takich modeli jest *K-SEIR* opisany w artykule *„K-SEIR-Sim: A simple customized software for simulating the spread of infectious diseases." [2]*

We wspomnianym artykule zaproponowany model rozszerza oryginalny SIR o dodatkową grupę *E - Exposed (narażeni)* oraz dodaje czynnik *K*, który określa działania przeciwdziałające zarazie podejmowane przez ludzi. Na podstawie modelu, dodatkowych parametrów oraz danych epidemiologicznych Covid-19 z miasta Wuhan zostały opracowane równania do matematycznego modelowania postępu rozprzestrzeniania się choroby, które autorzy przedstawili w tabeli.



Rysunek 2.1: Schemat działania modelu *K-SEIR*

Teoretyczny model K-SEIR został przekształcony w prosty oprogramowanie, zaimplementowane w języku PYTHON, przy użyciu inżynierii oprogramowania. W szczególności, ukończono zadania związane z projektowaniem interfejsu graficznego użytkownika, logiką sterowania, logiką operacji, kontrolą precyzji, kontrolą prędkości, wizualizacją danych, importem i eksportem danych, dopasowaniem parametrów, wyświetlaniem kluczowych danych oraz innymi konkretnymi treściami.

Tabela 2.1: Opis modelu epidemiologicznego *K-SEIR*.

Populacja	**Równanie**	**Parametry**
Podatni (S)	$\frac{ds}{dt} = -\frac{\lambda si}{N} + \mu h$	λ: średnia dzienna ilość zarażeń s: liczba populacji (S) w czasie t i: liczba populacji (I) w czasie t μ: średnia dzienna ilość ponownych zarażeń h: liczba populacji (H) w czasie t N: liczba całkowitej populacji w danym regionie
Narażeni (E)	$\frac{de}{dt} = \frac{\lambda si}{N} - \sigma e$	σ: wskaźnik zachorowań na dzień e: liczba populacji (E) w czasie t
Zarażeni (I)	$\frac{di}{dt} = \sigma e - \gamma i$	γ: średni dzienny współczynnik zmniejszania grupy zarażonych pacjentów
Usunięci (R)	$\frac{dr}{dt} = \gamma i$	Suma wyleczonych i zmarłych
Ozdrowieńcy (H)	$h = \alpha r$	r: liczba populacji (R) w czasie t α: średnia dzienna współczynnik zdrowienia
Zmarli (D)	$d = \beta r$	β: średnia dzienny współczynnik śmiertelności
	$\alpha + \beta = \gamma$	
	$s_0 + e_0 + i_0 + r_0 = N \quad (\text{dla } t = 0)$	0: czas t = 0
	$\lambda_k = (1 - k_1)\lambda$ $\gamma_k = k_2\gamma$ $\alpha_k = k_3\alpha$	K: współczynnik interwencji ludzkiej k_1: miara izolacji fizycznej, współczynnik λ k_2: zdolność przyjęcia do szpitala, współczynnik γ k_3: zdolność leczenia, współczynnik α

2.2 Modele agentowe

Modele agentowe stanowią komputerowe symulacje, w których agenci, reprezentujący różne jednostki, mogą wejść w interakcję między sobą. Agentami mogą być jednostki takie jak osoby, organizacje, a nawet obszary geograficzne, takie jak województwa czy kraje. Wzajemne oddziaływania między agentami są określone przez zdefiniowane zasady progra-

mowe. Każdy z agentów podejmuje decyzje indywidualnie, co może obejmować zarówno proste wybory, na przykład decyzję o kierunku ruchu, jak i bardziej złożone decyzje, takie jak znalezienie konkretnego innego agenta lub sekwencję zdarzeń. W kontekście symulacji zarażeń, tego typu podejście pozwala na realistyczne uwzględnienie nieprzewidywalnych decyzji, jakie może podjąć jednostka.

Jednym z artykułów naukowych, poruszających temat modelowania agentowego pod tytułem: *„An open-data-driven agent-based model to simulate infectious disease outbreaks"* [3] dostarczy nam cennych informacji na temat implementacji takich rozwiązań.

W artykule możemy przeczytać szczegółowe informacje na temat tego co badacze wzięli po uwagę podczas tworzenia swojego modelu. był on oparty o dane populacji, umiejscowienia szkół i miejsc pracy a także danych dotyczących szczepień. Rzeczywiste dane są używane do określenia struktury wiekowej i płciowej naszych populacji, wraz z właściwym rozkładem wielkości gospodarstw domowych oraz innymi cechami takimi jak wiek dzieci. Następnie dane opisujące lokalizację szkół oraz miejsc pracy dają możliwość wiernego odwzorowania interakcji pomiędzy ludźmi. Na koniec aby ocenić podatność populacji na rozprzestrzenianie się choroby uwzględniono ilość szczepień (badanie opierało się na badaniu epidemii odry). Następnym krokiem badaczy było wybranie testowanego obszaru i podzielenie go na mniejsze jednostki, w których przebywający ludzie są uznawani za mający kontakt ze sobą. Później należało rozmieścić agentów w odpowiednich domach według danych o populacji. Na koniec należało uwzględnić dodatkowe czynniki między innymi transport. Ostatnim krokiem było matematyczne opisane szans na zarażeniem, w tym celu autorzy wykorzystali równanie:

$$R_0 = cpd$$

gdzie:
R_0 - prawdopodobieństwo infekcji.
c - liczba kontaktów na jednostkę czasu.
p - szansa na zarażenie podczas kontaktu.
d - długość trwania infekcji

Poprzez przekształcenie równania otrzymano:

$$p = \frac{R_0}{cd}$$

Dodatkowo w scenariuszach testowych, w których brano pod uwagę szczepienia użyto dodatkowego równania. Jest ono oparte na odporność zbiorowej jednak zmodyfikowane aby uwzględniać skuteczność szczepionki:

$$V_c = \frac{(1-\frac{1}{R_0})}{V_e}$$

gdzie:
V_c - objęcie szczepieniami populacji
V_e - skuteczność szczepionki.

Badacze przetestowali swój program na wielu miastach w Irlandii, ze względu na dostępne dane co do ich struktury oraz populacji. Dodatkowo model został porównany z rzeczywistymi danych dotyczącymi epidemii odry w Schull, Irlandia w 2012 roku. Ze względu na losowość w modelu efekty każdej z symulacji było nieco inne.

cyt. *„ (...) Średnia liczba zainfekowanych agentów w różnych próbach wyniosła 17, przy maksymalnej liczbie 90 zainfekowanych agentów w jednym przypadku. Dwadzieścia pięć procent prób skończyło się wybuchem, w którym więcej niż 30 agentów zostało zainfekowanych. Wyniki pokazują, że chociaż średnia dla wszystkich prób jest niższa niż liczba zainfekowanych w przypadku wybuchu w Schull, to liczba faktycznie zainfekowanych osób znajduje się w 75. centylu wyników modelu." (...)*[3]

Podsumowując, przewagą modeli agentowych nad modelami takimi jak SEIR jest uwzględnienie decyzji jednostki, przykładowo jeżeli mimo choroby osoba zdecyduje się pójść do szkoły lub pracy, ilość zarażeń wzrośnie lub odwrotnie jeżeli zdecyduje się zostać w domu, ilość zarażeń zmaleje, takie niuanse są wychwytywane dzięki skupieniu się na jednostce. Niestety takie podejście dużo gorzej radzi sobie w miarę zwiększania populacji, dlatego idealnie nadają się do symulacja epidemii w małych miastach ale nie w obrębie całego kraju lub świata.

2.3 Inne modele symulacji zarażeń

Ta część zwróci uwagę na dwie innowacyjne metody modelowania zarażeń, które stanowią wyjątkowe podejście w szerokim spektrum dostępnych metod. Ze względu na obecności różnorodnych podejść do modelowania zarażeń, skoncentrujemy się teraz na dwóch szczególnie interesujących metodach. Pierwszą z nich jest ambitny projekt symulacyjny oparty na algorytmach równoległych, korzystający z obszernych danych dotyczących populacji w Stanach Zjednoczonych. Drugą, szczególnie oryginalną, jest aplikacja wykorzystująca technologię rozszerzonej rzeczywistości do wizualizacji, jak wirus może utrzymywać się na różnych powierzchniach i w konsekwencji jak może się szerzyć. Przechodząc do analizy tych dwóch wyjątkowych podejść w celu zidentyfikowania ich specyficznych zalet.

2.3.1 EpiSimdemics - równoległy algorytm symulacji epidemii

„EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks" [1] Jest bardzo ambitnym projektem wykorzystującym do obliczeń równoległy algorytm do symulowania rozprzestrzeniania się zarażeń w dużych

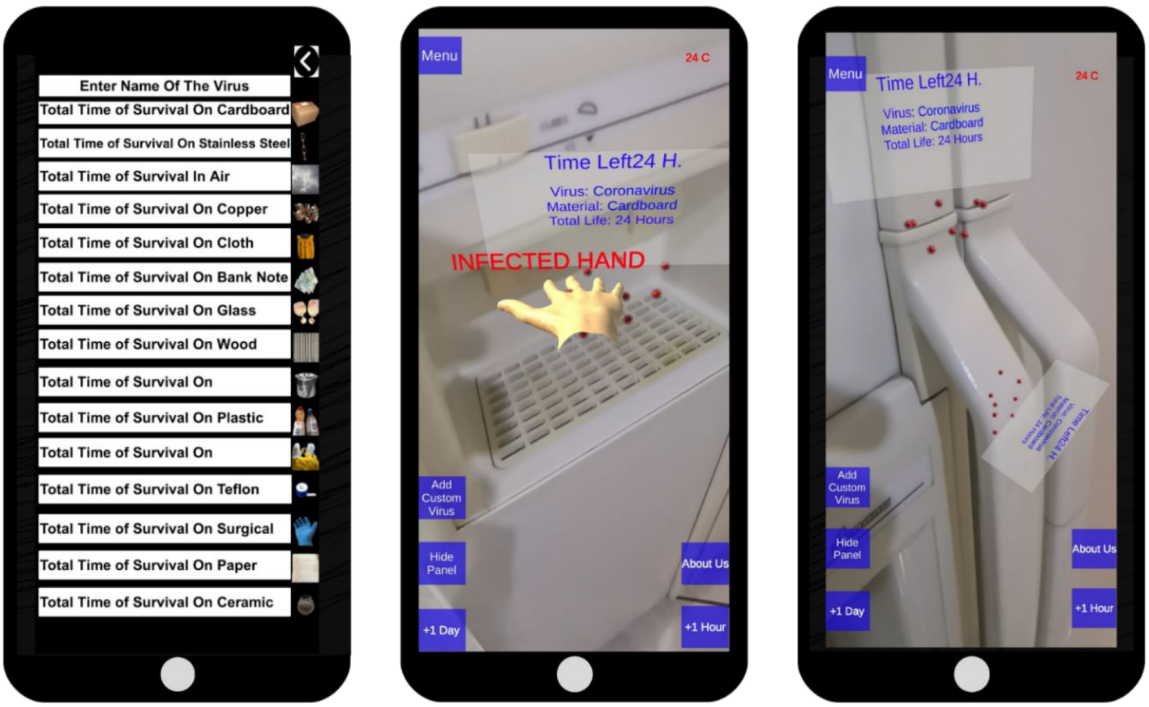
i realistycznych symulowanych społeczeństwach. Badacze postanowili zasymulować niemalże statystycznie nierozróżnialną populację Stanów Zjednoczonych. Każdy z agentów w symulacji jest inny i opisany przez nawet do 163 zmiennych demograficznych z spisu ludności. Algorytm *EpiSimdemics* opiera się na:

- kolekcji jednostek z wartościami stanu i lokalnych reguł przejść między stanami
- grafie interakcji przechwytującego lokalną zależność jednostki od swoich sąsiednich jednostek
- sekwencji aktualizacji lub harmonogramu, takiego że związek przyczynowo-skutkowy w systemie jest reprezentowany przez składanie lokalnych odwzorowań

Z tych założeń są formułowane równania przejść stanów dla każdego z osobników w symulacji. Reprezentujące w jaki sposób stan wierzchołka (osobnika) i jego sąsiadujących wierzchołków będzie zmieniał się w trakcie trwania programu. Innymi słowy definiuje proces rozprzestrzeniania się choroby w siatce wierzchołków reprezentujących społeczeństwo. Dzięki zastosowaniu tak skomplikowanego systemu, program *EpiSimdemics* posiada zdolność dostarczania szczegółowych informacji na temat rozprzestrzeniania się choroby w populacji, obejmujących takie detale jak konkretny zestaw osób zainfekowanych, miejsce zarażenia oraz kto ich zarażał.

2.3.2 Zastosowanie rozszerzonej rzeczywistości w symulacji zarażeń

Zainspirowani globalną pandemią COVID-19, badacze postanowili stworzyć innowacyjną aplikację pokazującą rozprzestrzenianie się wirusa poprzez różnego rodzaju powierzchnie i przedmioty, z którymi stykamy się w codziennym życiu. Udało się to dzięki wykorzystaniu rozszerzonej rzeczywistości, co pozwoliło na dokładne zobrazowanie, jak wirusy i bakterie mogą pozostawać na tych powierzchniach oraz jak łatwo mogą być przenoszone poprzez kontakty ręczne i inne interakcje. Aplikacja umożliwia użytkownikowi stworzenie własnego patogenu lub wybranie istniejącego, a następnie korzystając z kamery pozwala umieścić go w świecie wirtualnym. Użytkownik może śledzić, jak długo patogen utrzymuje się w danym miejscu, potencjalnie stanowiąc ryzyko przeniesienia się na inną osobę. Swoje spostrzeżenia i wnioski zawarli w artykule *Bio-Virus Spread Simulation in Real 3D Space using Augmented Reality*[4]



(a) (b) (c)

Rysunek 2.2: Demonstracja działania aplikacji: (a) dostosowywanie niestandardowego wirusa, (b) wprowadzenie wirusa w rzeczywistego świata, (c) wirus widoczny w rozszerzonej rzeczywistości z panelem informacyjnym.

Rozdział 3

Wymagania i narzędzia

Rozdział skupi się na omówieniu kluczowych aspektów dotyczących aplikacji do symulowania rozprzestrzeniania się zarażeń o nazwie *InfektoSym*. Rozpocznie od analizy i przedstawienia wymagań funkcjonalnych oraz niefunkcjonalnych. Następnie przybliży technologie wykorzystane w aplikacji. Dodatkowo, dokładnie wyjaśni działanie zaimplementowanego modelu symulacji rozprzestrzeniania się choroby, a także w jaki sposób aplikacja oddziałuje z użytkownikiem i jakie rezultaty może dostarczyć. Ten rozdział stanowi istotne wprowadzenie do zrozumienia zarówno technicznego, jak i funkcjonalnego aspektu projektu.

3.1 Wymagania funkcjonalne i niefunkcjonalne

3.1.1 Wymagania funkcjonalne

- **Symulacja Ruchu i Interakcji:**

  Aplikacja pozwala na śledzenie trajektorii ruchu każdej postaci w biurze w czasie rzeczywistym. Interakcje pomiędzy postaciami są symulowane z uwzględnieniem różnych scenariuszy, takich jak rozmowy, wspólna praca, czy przerwy. W momencie, gdy jedna postać zostaje zarażona, aplikacja monitoruje, czy i jak szybko choroba rozprzestrzenia się na inne postacie poprzez ich bezpośrednie kontakty.

- **Wizualizacja Stanów Zdrowia:**

  Na ekranie widoczne są dynamiczne wskaźniki zdrowia każdej postaci, pozwalające użytkownikowi śledzić ich aktualny stan (zdrowy, narażony, zarażony). Symulacja obejmuje także wizualizację okresów inkubacji oraz wyzdrowienia, umożliwiając obserwację zmian stanów zdrowia w czasie rzeczywistym.

- **Monitorowanie Kontaktów i Narażenia:**

Aplikacja zbiera dane dotyczące kontaktów pomiędzy postaciami, identyfikując te, które mogą prowadzić do potencjalnego zarażenia. Wizualizacja narażeń obejmuje różne aspekty, takie jak dystans, czas trwania kontaktu oraz ewentualne zastosowane środki ochrony osobistej.

- **Wariacje Scenariuszy:**

  Aplikacja umożliwia zmianę warunków symulacji, pozwalając użytkownikowi eksperymentować z różnymi scenariuszami zarażenia. Możliwość wprowadzania dynamicznych zmian w otoczeniu biurowym, takich jak nowe stanowiska pracy czy przeniesienie osób, dla lepszego zrozumienia wpływu organizacji przestrzeni na rozprzestrzenianie się infekcji.

- **Wysoce Parametryzowalna Symulacja:**

  1. Dystans Zarażenia:

     Użytkownik ma możliwość określenia maksymalnego dystansu, na jakim wirus może się przenosić między agentami.

  2. Czas do Zarażenia:

     Określenie czasu, jaki musi upłynąć w bliskim kontakcie z zarażoną postacią, aby doszło do zarażenia.

  3. Zaraźliwość Patogenu:

     Parametr definiujący zdolność wirusa do zarażania innych postaci w danym środowisku symulacyjnym.

  4. Średni Okres Inkubacji:

     Ustalenie czasu, jaki upływa od momentu zarażenia do pojawienia się objawów u zarażonej postaci.

  5. Procent Populacji Noszący Maseczki:

     Możliwość określenia odsetka populacji, który stosuje ochronę w postaci noszenia maseczek.

  6. Skuteczność Maseczek:

     Parametr określający, o ile procent zmniejsza się dystans zarażenia dla osób noszących maseczki.

  7. Liczebność Populacji:

     Określenie ogólnej liczby postaci uczestniczących w symulacji.

  8. Odporność Populacji:

     Ustalenie procenta populacji posiadającego naturalną odporność na wirusa.

  9. Długość Symulacji:

     Określenie czasu trwania symulacji w jednostkach czasu.

10. Prędkość Symulacji:

    Umożliwienie regulacji prędkości symulacji, włączając przyspieszenie do 100-krotności normalnej prędkości.

11. Pauzowanie Symulacji:

    Dodatkowa funkcjonalność, która pozwala na zatrzymywanie symulacji w dowolnym momencie i jej późniejsze wznowienie.

3.1.2 Wymagania niefunkcjonalne

- **Intuicyjny Interfejs:** Interfejs użytkownika powinien być zaprojektowany w sposób intuicyjny, umożliwiając łatwe poruszanie się po aplikacji i korzystanie z jej funkcji. Elementy graficzne, przyciski i opcje powinny być jasne i zrozumiałe dla użytkownika końcowego.

- **Stabilność:**

  Aplikacja powinna charakteryzować się stabilnością działania, eliminując nieoczekiwane błędy, które mogą prowadzić do awarii. Wszystkie funkcje aplikacji powinny działać zgodnie z oczekiwaniami, zapewniając płynne doświadczenie użytkownika.

- **Odporność na Awarie:**

  Aplikacja powinna być odporne na awarie poprzez implementację mechanizmów zabezpieczających, takich jak obsługa błędów, przywracanie stanu aplikacji po awarii, oraz minimalizacja wpływu awarii na całość systemu.

- **Płynność Symulacji:**

  Aplikacja powinna zapewniać płynną symulację nawet przy dużej ilości agentów uczestniczących w scenariuszu. Optymalizacje i zoptymalizowany kod powinny umożliwiać utrzymanie odpowiedniej prędkości symulacji, niezależnie od skomplikowania scenariusza.

- **Responsywność Interfejsu:**

  Interfejs użytkownika powinien reagować szybko na akcje użytkownika, zapewniając natychmiastowe odpowiedzi na interakcje, co przyczyni się do lepszego doświadczenia użytkownika.

3.2 Wybrana technologia

Rozważając wybór technologii do stworzenia aplikacji *InfektoSym*, zdecydowano się na silnik gier Unity. Ten wybór był podyktowany kilkoma kluczowymi czynnikami, mającymi istotne znaczenie dla skuteczności i efektywności projektu.

Przede wszystkim, istnienie dużej społeczności użytkowników stanowiło istotny argument. Unity cieszy się uznaniem ze względu na szeroki dostęp do materiałów szkoleniowych wideo, aktywność na forum dyskusyjnych oraz obfite źródła dokumentacji online. Ta społeczność stanowi nie tylko źródło wsparcia, lecz także umożliwia szybsze rozwiązywanie potencjalnych problemów napotkanych podczas procesu tworzenia aplikacji.

Duże możliwości rozwoju były kolejnym czynnikiem decydującym o wyborze Unity. Silnik ten oferuje elastyczność i skalowalność, co pozwala na rozbudowę aplikacji w miarę ewentualnych zmian w wymaganiach projektu.

Aspekt wydajności miał kluczowe znaczenie dla płynności symulacji. Unity, dzięki zoptymalizowanym mechanizmom renderowania i obsługi fizyki - pierwotnie zaprojektowanych do tworzenia gier komputerowych, świetnie odnajduj się w przeprowadzaniu wszelkiego rodzaju symulacji gdzie ważna jest reprezentacja graficzna.

Gotowe narzędzia dostarczane przez Unity, takie jak edytory interfejsu użytkownika czy wbudowany system nawigacji dla poruszających się obiektów, znacząco skracają czas potrzebny na rozwój aplikacji. To ułatwienie pozwala skupić się na kluczowych aspektach projektu.

Podsumowując, wybór Unity jako platformy do tworzenia *InfektoSym* wynikał z korzyści płynących z bogatego ekosystemu, wydajności silnika oraz dostępności narzędzi ułatwiających pracę, co zapewniło efektywny i skuteczny proces realizacji projektu.

3.3 Model symulacji rozprzestrzeniania się zarażeń zastosowany w InfektoSym

Model symulacji zaimplementowany w *InfektoSym* stanowi połączenie modelu SEIR z podejściem opartym na agentach, mając na celu dokładne odwzorowanie rozprzestrzeniania się zarażeń. Struktura modelu zakłada podział populacji na cztery główne grupy, analogiczne do SEIR:

- **Zdrowi (Susceptible)**
- **Narażeni (Exposed)** - grupa osób, które miały kontakt z zarażonym i są potencjalnie podatne na zakażenie.
- **Zarażeni (Infected)** - grupa osób, u których choroba jest aktywna, co stanowi źródło dalszego zarażania.
- **Usunięci (Removed)** - osoby, które zostały zidentyfikowane jako chore i zostały izolowane.

Symulacja opiera się na podejściu agentowym, skupiając uwagę na indywidualnych decyzjach każdego z osobników. Ten szczególny nacisk na indywidualność umożliwia uchwyce-

nie subtelności wprowadzanych przez decyzje jednostki, co jest istotne dla precyzyjnego modelowania dynamiki rozprzestrzeniania się zarażeń.

3.3.1 Równania opisujące model symulacji

Założenia modelu oraz parametry symulacji pozwalają wyizolować równania opisujące prawdopodobieństwo zarażenia w określonych warunkach.

Przejście agenta ze stanu zdrowego do narażonego obliczane jest według następujących parametrów i równań:

Jeżeli zdrowa osoba przebywa w odległości $x < d_{maks}$, gdzie x to odległość od zarażonego, a d to maksymalny dystans, na jaki patogen może się przenosić w czasie $t_k > t_z$, gdzie t_k to czas kontaktu, a t_z to minimalny czas potrzebny do zakażenia. Szansa na przeniesienie do grupy narażonych (exposed) określana jest przez parametr zdolności zarażania patogenu R_0 (jeżeli $R_0 = 50$, przy każdym kontakcie szansa na narażenie wynosi 50%). Dodatkowymi parametrami grającymi tu rolę jest posiadanie maseczki i jej skuteczność.

Kiedy agent zostanie uznany za narażonego, program ponownie wylicza procentowe szanse na rozwój choroby, czyli zostanie zarażonym.

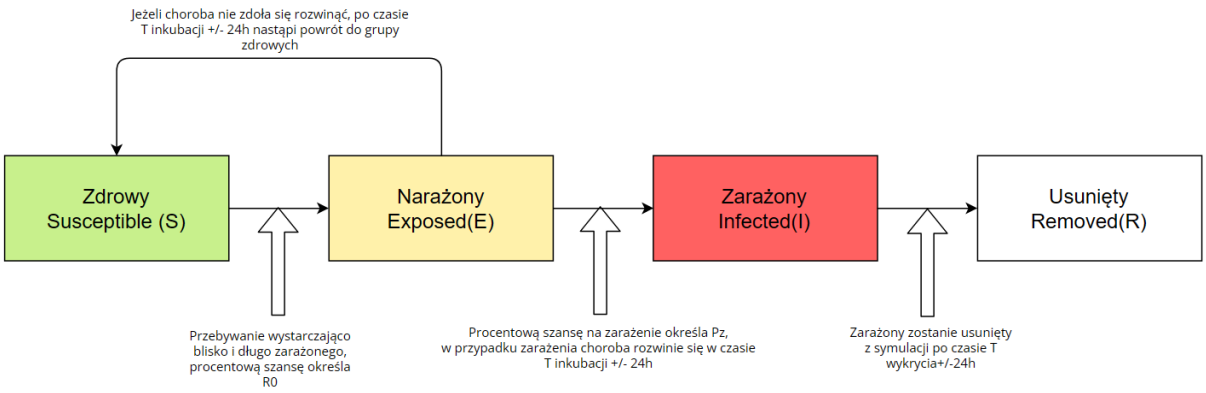
$$P_z = R_0 \cdot (1 - I)$$

gdzie:
$P_z \in <0, 100>$ - szansa na zostanie zarażonym, wyrażona w %
$R_0 \in <0, 100>$ - współczynnik zakażalności patogenu
$I \in <0, 1>$ - odporność jednostki.

Wykonywane jest losowanie; jeżeli tak, osobnik zostanie przeniesiony do grupy zarażonych (Infected), jeżeli nie, wróci do puli zdrowych (Susceptible). W obu przypadkach stanie się to po czasie $T_{inkubacji} \pm 24$ godziny (czasu symulacji).

Osoba zakażona jest usuwana z symulacji po czasie $T_{wykrycia} \pm 24$ godziny (czasu symulacji).

3.3.2 Symulacja zachowań ludzkich



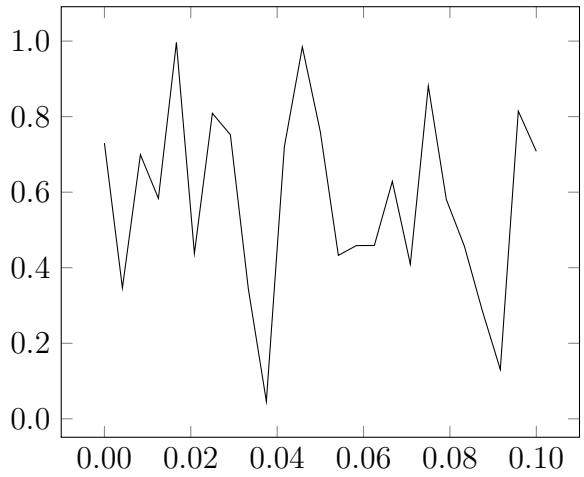
Rysunek 3.1: Schemat działania modelu użytego w *InfektoSym*

Rozdział 4

[Właściwy dla kierunku – np. Specyfikacja zewnętrzna]

Jeśli „Specyfikacja zewnętrzna”:

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)



Rysunek 4.1: Podpis rysunku po rysunkiem.

Rozdział 5

[Właściwy dla kierunku – np. Specyfikacja wewnętrzna]

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótka wstawka kodu w linii tekstu jest możliwa, np. `int a;` (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.1, a naprawdę długie fragmenty – w załączniku.

```
1  class test : public basic
2  {
3      public:
4        test (int a);
5        friend std::ostream operator<<(std::ostream & s,
6                                       const test & t);
7      protected:
8        int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w `listings`.

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

	metoda						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna . . .)
- problemy napotkane w trakcie pracy

Bibliografia

[1] Chris Barrett, Keith Bisset, Stephen Eubank, Xizhou Feng i Madhav Marathe. „EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks". W: sty. 2008, s. 37. DOI: 10.1145/1413370.1413408.

[2] Chaobao Zhang Xiaona Wei Xiangqi Li Hongzhi Wanga Zhiying Miao. „K-SEIR-Sim: A simple customized software for simulating the spread of infectious diseases". W: *Computational and Structural Biotechnology Journal* 19 (2021), s. 1966–1975.

[3] Elizabeth Hunter, Brian Mac Namee i John Kelleher. „An open-data-driven agent-based model to simulate infectious disease outbreaks". W: *PLOS ONE* 13.12 (grud. 2018), s. 1–35. DOI: 10.1371/journal.pone.0208775. URL: https://doi.org/10.1371/journal.pone.0208775.

[4] Kostandinos Tsaramirsis, Akshet Patel, Pranav Sharma, Nikunj Polasani, Princy Randhawa, Georgios Tsaramirsis, Athanasia Pavlopoulou, Zeynep Kocer i Dimitrios Piromalis. „Bio-Virus Spread Simulation in Real 3D Space using Augmented Reality". W: *Engineered Science* (sty. 2021). DOI: 10.30919/es8d592.

Dodatki

Spis skrótów i symboli

SIR Suceptible, Infected, Removed (podatni-zainfekowani-ozdrowieńcy)

SEIR Suceptible, Exposed, Infected, Removed (podatni-wystawieni-zainfekowani-ozdrowieńcy)

MVC model – widok – kontroler (ang. *model–view–controller*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

$\mathbb{E}$ zbiór krawędzi grafu

$\mathcal{L}$ transformata Laplace'a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
          iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
          epsilon set — you should set either number of iterations
          or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel