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 2024

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 [4].

Dziedzina informatyki w znacznym stopniu przyczynia się do rozwiązania problemów związanych z pandemią, dowodem na to jest niezliczona ilość artykułów naukowych związanych z tym tematem opublikowanych w czasie pandemii [10][1][6]. 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. Dodatkowo, umożliwiają szybkie testowanie różnych scenariuszy i strategii reakcji na sytuacje kryzysowe, co przyczynia się do skuteczniejszego przygotowania się do potencjalnych zagrożeń zdrowotnych [3].

Niniejsza 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.

Pracę podzielono na siedem rozdziałów. Na początku pracy dokonano analizy ist-

niejących modeli symulacji zarażeń. Następnie sprecyzowano wymagania projektowe i narzędzia, które posłużyły do realizacji projektu. W dalszej części opisano specyfikacje wewnętrzną i zewnętrzną programu oraz scenariusze dotyczące walidacji i testowania. Na końcu przedstawiono wnioski i dalsze możliwości rozwoju aplikacji.

Rozdział 2

Modele symulacji rozprzestrzeniania się zarażeń

Symulacje komputerowe są niezwykle istotnym narzędziem badawczym, umożliwiającym przygotowanie się do różnych sytuacji kryzysowych oraz testowanie scenariuszy. To przyczynia się do lepszego zrozumienia i skutecznego zarządzania w przypadku wystąpienia lokalnych epidemii, co może przeciwdziałać ich przerodzeniu się w globalną pandemię.

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 jeden. 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. 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 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.

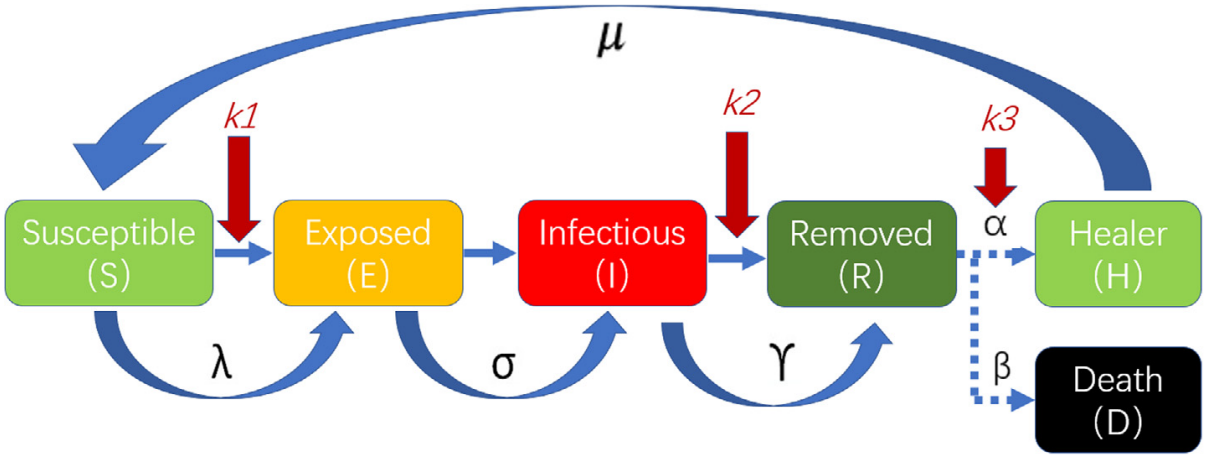
Wzmożone prace w dziedzinie modelowania epidemiologicznego miały miejsce na początku XX wieku, przez takich badaczy jak Ronald Ross[15] i Hilda Phoebe Hudson[13][14]. Świat po I wojnie światowej stanął przed pandemią grypy hiszpanki, która zarażając 1/3 ówczesnej populacji i powodując więcej ofiar niż dopiero co zakończony globalny konflikt

zbrojny[5], spowodowała pilną potrzebę zrozumienia i kontrolowania takich masowych zjawisk. W odpowiedzi na konieczność zbadania dynamiki pandemii grypy hiszpanki, powstało wiele matematycznych modeli symulacyjnych dotyczących rozprzestrzeniania się chorób zakaźnych, na przykład teoria Kermacka-McKendricka[17] uważaną jako podstawę modelu SIR (podatni-zainfekowani-ozdrowieńcy). Model ten, 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* [8] .

We wspomniany 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 tabeli2.1.



Rysunek 2.1: Schemat działania modelu *K-SEIR*[8]

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* opracowany na podstawie[8].

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.

Jeden z artykułów naukowych, poruszający temat modelowania agentowego, pod tytułem: *„An open-data-driven agent-based model to simulate infectious disease outbreaks"* [9] 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." (...)* [9]

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 aplikacje do symulacji zarażeń

W kontekście licznych podejść do modelowania zarażeń, niemożliwe jest przedstawienie wszystkich dostępnych rozwiązań. W tej sekcji skupiono się zatem na przedstawieniu dwóch szczególnie ciekawych metod oraz analizie innych aplikacji dostępnych na rynku, co pozwoli na pełniejsze zrozumienie różnorodności narzędzi do symulacji epidemii. Poprzez analizę tych dwóch podejść oraz eksplorację rynkowych rozwiązań, będziemy mogli lepiej zidentyfikować specyficzne zalety i wartość każdej z badanych metod w kontekście modelowania zarażeń.

2.3.1 EpiSimdemics - równoległy algorytm symulacji epidemii

„EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks" [2] 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

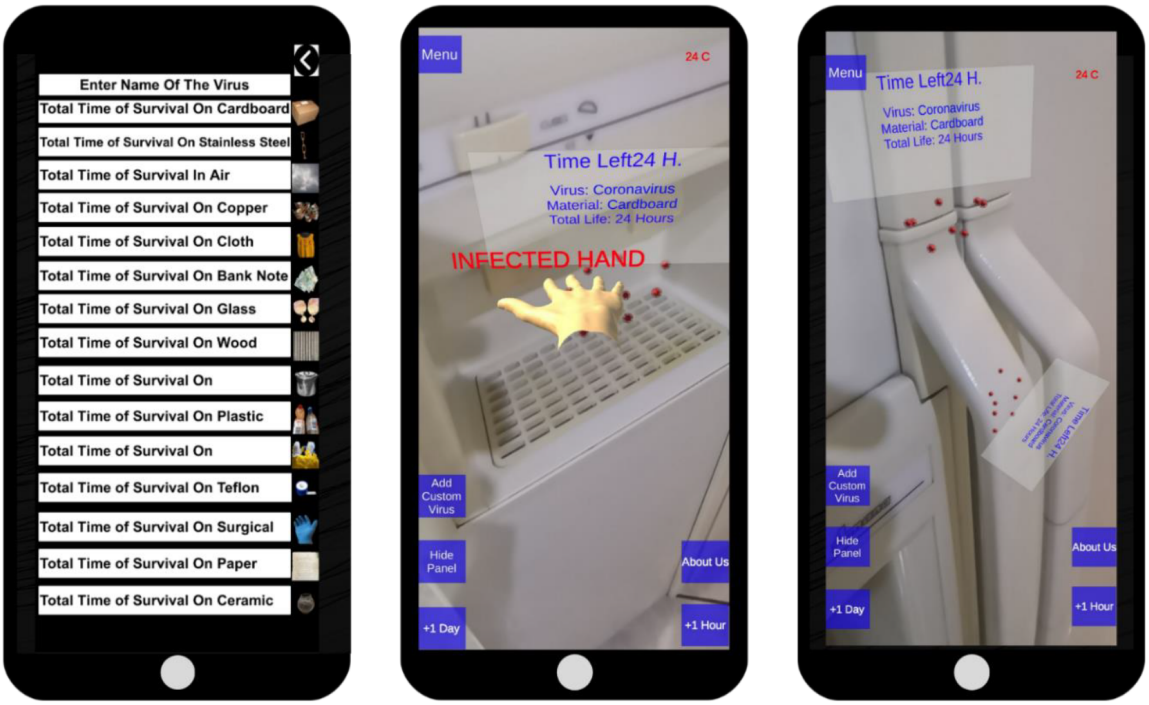
Z tych założeń są formułowane równania przejść stanów dla każdego z osobników w symulacji. Reprezentujące w jaki sposób stan wierzchołka (osobnika) i jego sąsiadujących wierzchołków będzie zmieniał się w trakcie trwania programu. Innymi słowy definiuje proces rozprzestrzeniania się choroby w siatce wierzchołków 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 2.2. 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 badacze zawarli w artykule *Bio-Virus Spread Simulation in Real 3D Space using Augmented Reality* [16]

2.3.3 Aplikacje dostępne w internecie i sklepach z aplikacjami

W dzisiejszym świecie aplikacji internetowych oraz sklepów z aplikacjami można znaleźć różnorodne narzędzia związane z modelowaniem i symulacją zarażeń. Jednym z przykładów jest aplikacja stworzona przez Biointeractive *„Modeling Disease Spread”* [12],

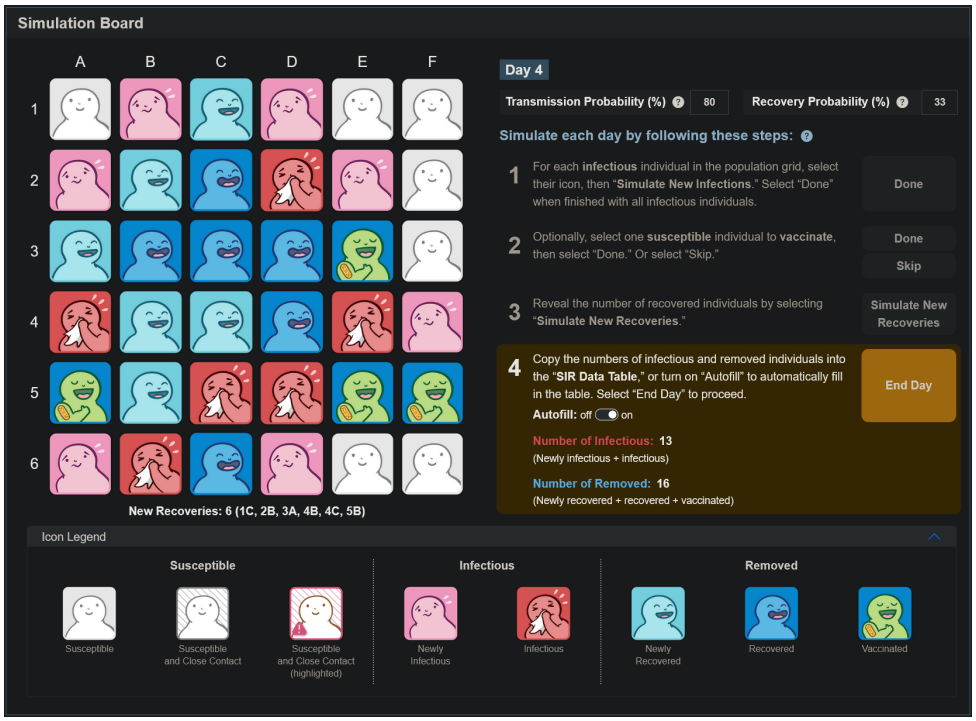


(a) (b) (c)

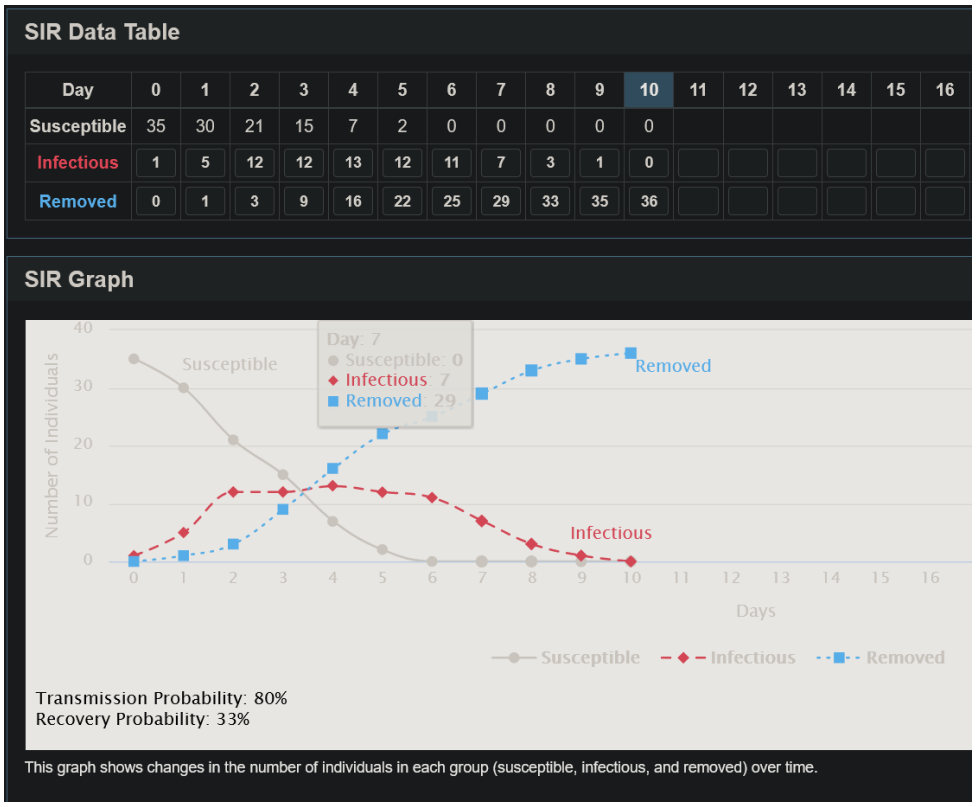
Rysunek 2.2: Demonstracja działania aplikacji: (a) dostosowywanie niestandardowego wirusa, (b) wprowadzenie wirusa w rzeczywistego świata, (c) wirus widoczny w rozszerzonej rzeczywistości z panelem informacyjnym [16].

która umożliwia interaktywne śledzenie modelu SIR. Warto jednak zauważyć, że ta konkretna aplikacja, skoncentrowana głównie na prezentacji wykresów i danych, jest raczej statycznym narzędziem 2.3.

Przejście do sklepów z aplikacjami, takich jak Google Play, odsłania kolejny aspekt tego tematu. Choć istnieje wiele aplikacji związanych z modelowaniem zarażeń, niestety większość z nich przybiera formę gier, traktując temat bardziej jako formę rozrywki niż naukowego czy edukacyjnego narzędzia [7] [11]. Te aplikacje często skupiają się na abstrakcyjnych czy fantastycznych scenariuszach, mając niewiele wspólnego z realistycznym oddaniem symulacji zarażeń. W efekcie istnieje wyraźna potrzeba bardziej zaawansowanych i edukacyjnych narzędzi, które mogą rzetelnie modelować i symulować procesy związane z rozprzestrzenianiem się chorób zakaźnych.



(a) Interfejs aplikacji [12].



(b) Statystyki generowane przez aplikację [12].

Rysunek 2.3: Wygląd aplikacji *„Modeling Disease Spread”* [12].

Rozdział 3

Wymagania i narzędzia

Rozdział omawia kluczowe aspekty dotyczące aplikacji do symulowania rozprzestrzeniania się zarażeń o nazwie *InfektoSym*. Rozpocznie się 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 eksperymentowanie z różnymi scenariuszami zarażenia poprzez dostosowanie parametrów symulacji. Użytkownik może modyfikować takie czynniki jak dystans zarażenia, czas do zarażenia, czy skuteczność maseczek, co pozwala na badanie wpływu tych parametrów na proces rozprzestrzeniania się infekcji w przestrzeni biurowej.

- **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. **Początkowy Procent Zarażonych:**

     Określenie procentowej liczby zarażonych w początkowej populacji.

  9. **Odporność Populacji:**

     Ustalenie procenta populacji posiadającego naturalną odporność na wirusa.

10. **Długość Symulacji:**

    Określenie czasu trwania symulacji w jednostkach czasu.

11. **Prędkość Symulacji:**

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

12. **Średni czas wykrycia:**

    Ustalenie czasu, jaki upływa od momentu pojawienia się objawów, do ich wykrycia i zlecenia kwarantanny.

13. **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 odnajduje 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)** - grupa osób zdrowych.
- **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 odizolowane.

Symulacja opiera się na podejściu agentowym, skupiając uwagę na indywidualnych decyzjach każdego z osobników. Ten szczególny nacisk na indywidualność umożliwia uchwycenie 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. Dystans przenoszenia się patogenu jest procentowo zmniejszany w zależności od ustawionej skuteczności maseczek.

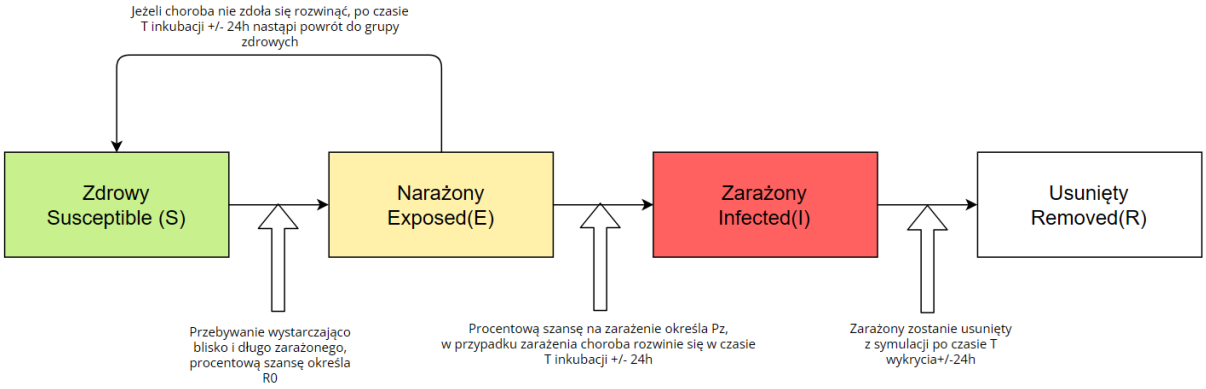
Kiedy agent zostanie uznany za narażonego, program wylicza procentowe szanse na rozwój choroby. Na podstawie wzoru:

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

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).



Rysunek 3.1: Schemat działania modelu użytego w *InfektoSym*

3.3.2 Symulacja zachowań ludzkich

W ramach symulacji zachowań ludzkich w aplikacji, agenci mają zdefiniowane konkretne czynności, które mogą wykonywać. Poniżej przedstawiono proces realizacji tych aktywności (poniższe wartości czasowe odnoszą się do czasu symulacji):

- **Początkowe Losowanie:**
  - Kiedy agent pojawiając się w symulacji, losuje jedną z czterech podstawowych akcji: losowe spacerowanie, praca, przerwa lub lunch.
  - Każda akcja(oprócz losowego spacerowania) trwa od 1 do 4 godzin.
- **Zmiana Akcji:**
  - Po zakończeniu aktualnej akcji, agent wybiera nową spośród losowego spacerowania, pracy, przerwy lub lunchu.
  - Ten proces powtarza się, co pozwala agentowi na cykliczne zmiany działań.
- **Rozmowy:**
  - Kiedy agenci się mijają, mają 1% szansy na rozpoczęcie rozmowy.
  - Rozmowa trwa od 10 minut do 1 godziny, a po jej zakończeniu agent wybiera nową akcję.
- **Praca, Odpoczynek i Lunch:**
  - Jeśli agent zdecyduje się pracować, odpoczywać lub coś zjeść sprawdza dostępność odpowiednio biurek, miejsc na kanapie, stolików w kuchni.
  - Siada przy pierwszym wolnym miejscu i wykonuje czynność od 1 do 4 godzin, po czym wybiera nową akcję.
  - W przypadku braku wolnych miejsc agent rozpoczyna losowe spacerowanie.

Ten proces symulacyjny pozwala na odwzorowanie różnorodnych działań agentów, obejmujących pracę, odpoczynek, jedzenie i społeczne interakcje. Losowanie czasu trwania i zmiana akcji wprowadza naturalność w zachowaniach agentów.

Rozdział 4

Specyfikacja zewnętrzna

4.1 Opis Aplikacji

Aplikacja została stworzona w środowisku Unity i zapewnia interaktywną symulację rozprzestrzeniania się zarażeń w przestrzeni biurowej. Pozwala na zmianę wielu parametrów wpływających na przebieg symulacji. Ma na celu w prosty i zrozumiały dla każdego sposób pokazywać rozprzestrzenianie się choroby. Przeznaczona jest głównie dla użytkowników korzystających z komputerów osobistych z systemem operacyjnym Windows.

4.2 Wymagania Sprzętowe

- Komputer z systemem operacyjnym Windows.
- Ekran o rozdzielczości co najmniej 1280x720 pikseli.
- Karta graficzna wspierająca OpenGL 3.2 lub nowszy.

4.3 Wymagania Programowe

- System operacyjny: Windows 7/8/10 i nowsze.
- Zainstalowany runtime Unity w wersji zgodnej z aplikacją.

4.4 Obsługa Aplikacji

Do obsługi aplikacji wystarczy myszka, która umożliwia interakcję z interfejsem graficznym. Aplikacja nie wymaga dodatkowego sprzętu.

- **Ustawianie Parametrów Symulacji**
  odbywa się poprzez wybranie odpowiednich wartości na suwakach.

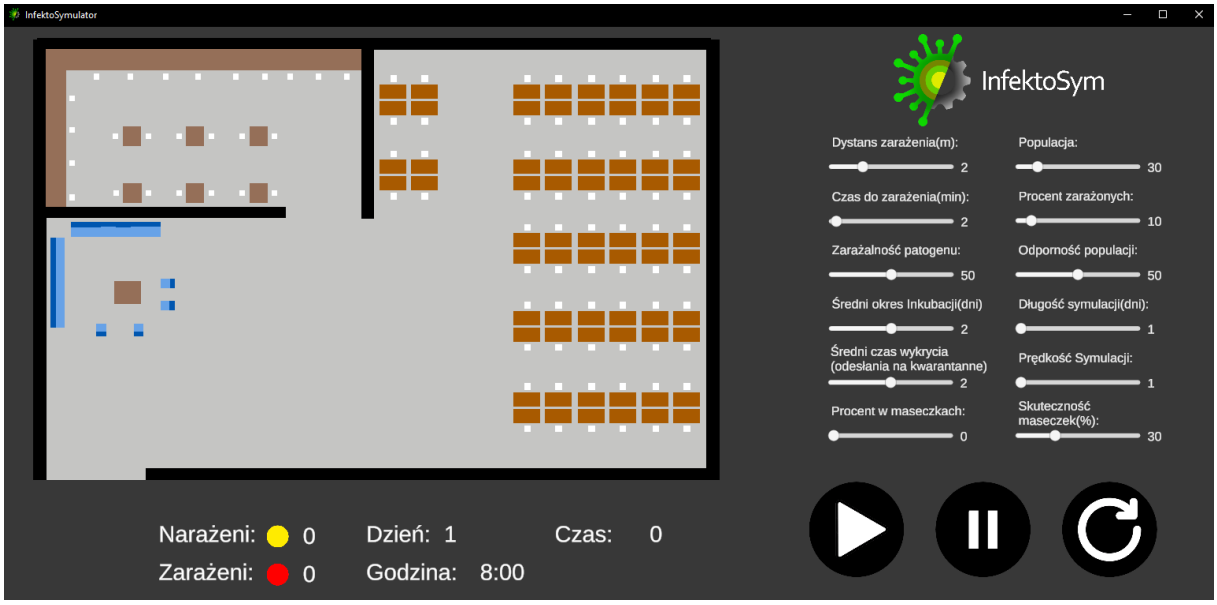
- **Rozpoczęcie symulacji i jej kontrola**
  aby rozpocząć symulacje należy po ustawieniu parametrów wcisnąć przycisk. W trakcie trwania symulacji jej prędkość można ustawiać odpowiednim suwakiem. Aby zatrzymać symulację należy wcisnąć przycisk pauzy. Aby zrestartować symulację należy wcisnąć przycisk restartu.

4.5 Uruchamianie Aplikacji

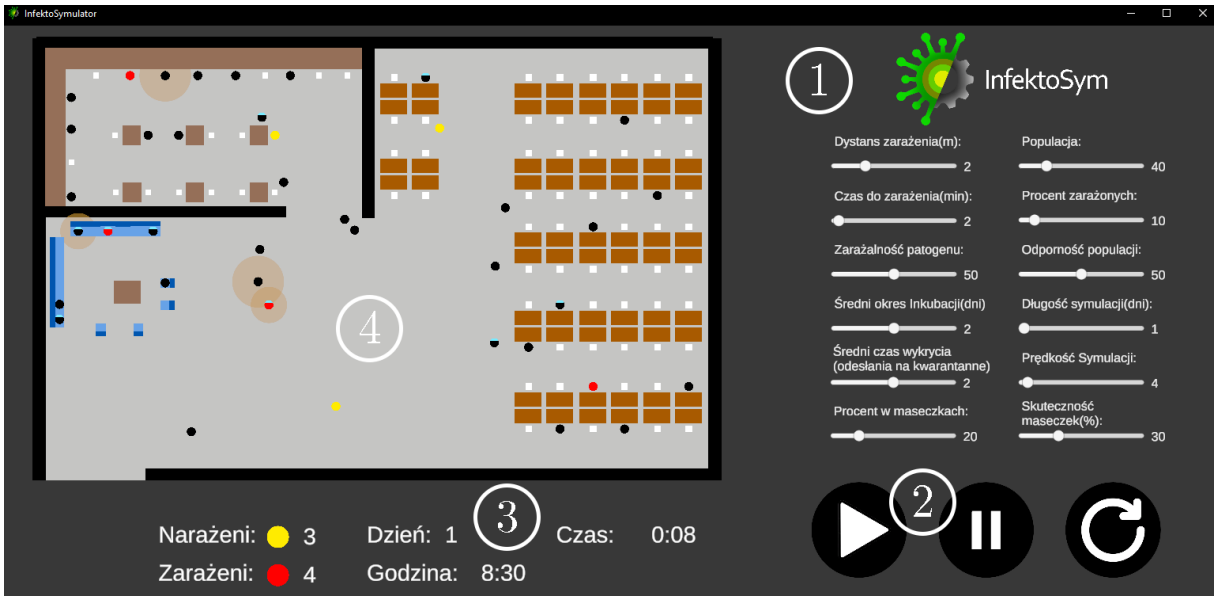
Aby skorzystać z aplikacji, należy wykonać poniższe kroki:

1. Pobrać folder zawierający aplikację.
2. Rozpakować pobrany folder w wybranym miejscu na dysku.
3. Kliknąć w folder aplikacji.
4. Kliknąć dwukrotnie plik wykonywalny (exe) aplikacji.

Aplikacja uruchomi się bez konieczności instalacji.

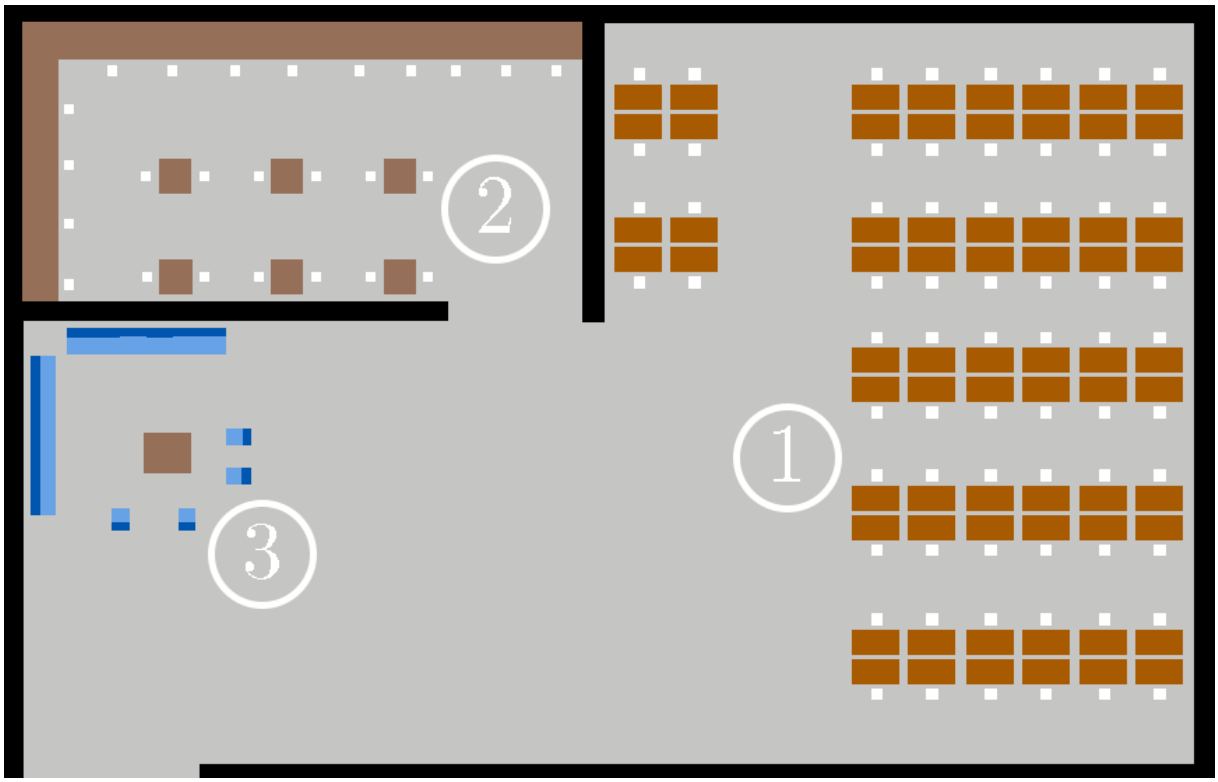


Rysunek 4.1: Ekran początkowy aplikacji.



Rysunek 4.2: Ekran podczas działającej symulacji: 1. Suwaki do ustawiania parametrów, 2. Przyciski kontrolujące 3. Aktualne statystyki i czas, 4. Wizualizacja

Rysunek 4.3: Ekran zakończenia symulacji



Rysunek 4.4: Mapa biura: 1. Biurka do pracy, 2. Kuchnia ze stolikami, 3. Miejsce wypoczynku

Rozdział 5

Specyfikacja wewnętrzna

W niniejszym rozdziale Specyfikacji Wewnętrznej dokładnie omówione zostaną kluczowe elementy związane z implementacją projektu. Zaprezentowana będzie architektura systemu, opisane zostaną użyte biblioteki i moduły, przedstawione zastosowane algorytmy, a także zamieszczone zostaną diagramy sekwencyjne, fragmenty kodu oraz idea, która kierowała procesem tworzenia aplikacji. Celem tego rozdziału jest szczegółowe przybliżenie struktury wewnętrznej projektu, zrozumienie jego komponentów oraz logiki działania.

5.1 Przedstawienie idei

Aplikacja Infektosym jest narzędziem symulacyjnym, które zostało stworzone z myślą o zrozumieniu i analizie rozprzestrzeniania się zarażeń w przestrzeni biurowej. Głównym celem projektu jest dostarczenie interaktywnej platformy umożliwiającej obserwację, analizę oraz testowanie scenariuszy związanych z potencjalnymi wirusowymi infekcjami w środowisku pracy.

- **Cele Projektu:**
  - Zapewnienie realistycznego modelu symulacji, uwzględniającego codzienne zachowania ludzkie w biurze.
  - Możliwość dostosowywania parametrów symulacji, takich jak dystans zarażenia, czas do zarażenia czy skuteczność maseczek.
  - Wizualizacja procesu rozprzestrzeniania się patogenów w czasie rzeczywistym.
- **Koncepcje i Założenia:**
  - Implementacja agentowego podejścia, gdzie każdy osobnik w symulacji podejmuje indywidualne decyzje i reaguje na otoczenie.
  - Duża parametryzacja symulacji, umożliwiająca dostosowanie scenariuszy do różnych warunków i kontekstów.

- **Kierunki Rozwoju:**
  - Wprowadzenie zaawansowanych funkcji interakcji między agentami, uwzględniających bardziej złożone scenariusze zachowań.
  - Integracja z danymi epidemiologicznymi dla bardziej precyzyjnych analiz i prognoz.
  - Dalsza optymalizacja interfejsu użytkownika i dostępność na różnych platformach.
  - Wprowadzenie innych scenariuszy np. Szpital, Osiedle, Centrum Handlowe.

5.2 Architektura systemu

Architektura systemu aplikacji opiera się na modularnym podejściu, skupiającym się na kilku kluczowych skryptach odpowiedzialnych za różne aspekty symulacji.

- **InterfaceScript:**
  - Odpowiada za interakcję z użytkownikiem, umożliwiając mu dostosowywanie parametrów symulacji za pomocą sliderów.
  - Obsługuje przyciski kontroli, takie jak start, pauza, restart, co zapewnia płynną kontrolę nad symulacją.
  - Wyświetla istotne statystyki dotyczące aktualnego stanu symulacji, umożliwiając użytkownikowi bieżącą analizę danych.
- **GlobalClockScript:**
  - Pełni rolę zegara w aplikacji, monitorując czas zarówno w skali symulacji, jak i rzeczywistości.
  - Liczy dni, godziny oraz ilość godzin symulacji od jej rozpoczęcia, co pozwala na śledzenie postępu w czasie.
  - Zapewnia synchronizację między symulacją a realnym czasem, co jest istotne dla precyzyjnego odwzorowania dynamiki zarażeń.
- **HumanSpawner:**
  - Inicjalizuje agentów (ludzi) na mapie, ustawiając je zgodnie z parametrami otrzymanymi od InterfaceScript podczas rozpoczęcia symulacji.
  - Zapewnia jednolite warunki startowe dla agentów, co umożliwia kontrolowaną analizę scenariuszy symulacyjnych.
- **HumanScript:**

  - Stanowi rdzeń symulacji zachowań ludzkich, przypisany do każdego obiektu reprezentującego osobę na mapie.
  - Monitoruje kolizje i stosuje algorytmy określające, czy dany osobnik powinien być narażony na ryzyko zarażenia czy też jest już zainfekowany.
  - Umożliwia kompleksową symulację interakcji między ludźmi i symulowanie ich zachowań.

- **SeatScript:**
  - Prosty skrypt odpowiedzialny za zarządzanie miejscami siedzącymi, informując, czy dane miejsce jest zajęte czy wolne.

5.2.1 Biblioteki i moduły

Aplikacja została napisana w Unity, wykorzystując standardowe biblioteki i moduły dostępne w tym środowisku. Dodatkowo, do realizacji funkcjonalności związanych z nawigacją postaci została użyta biblioteka Navmesh 2D. Navmesh to technika używana w grach do generowania trójwymiarowych lub dwuwymiarowych map nawigacyjnych, pozwalających postaciom na inteligentne poruszanie się w przestrzeni, omijanie przeszkód i podejmowanie decyzji dotyczących ruchu.

5.3 Algorytmy

W projekcie występuje kilka algorytmów kontrolujących symulację, zajmujących się kontrolą czasu, cyklem dnia i nocy, symulacją przenoszenia się choroby oraz symulacją ludzkich zachowań. Poniżej przedstawiono opis dwóch kluczowych algorytmów:

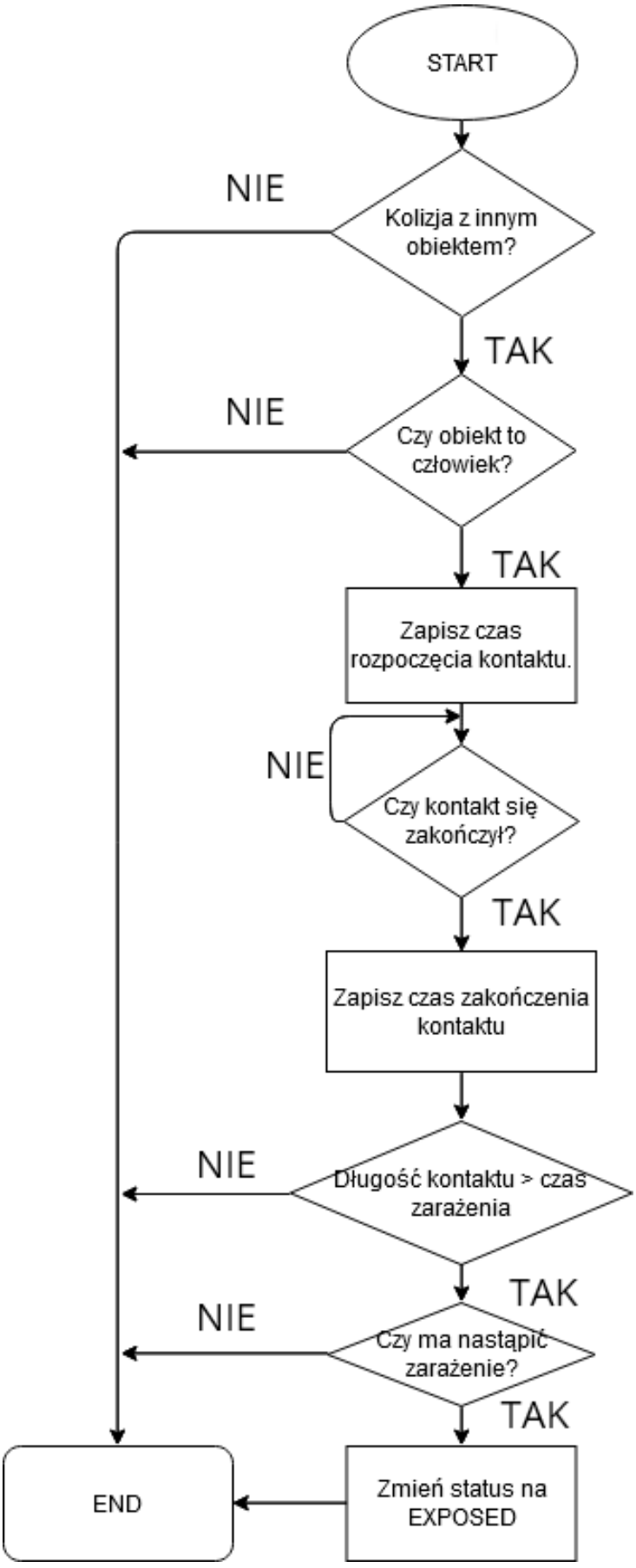
5.3.1 Kontrola rozprzestrzeniania się choroby

Algorytm ten skupia się na precyzyjnej symulacji interakcji między osobami. W przypadku kolizji między dwiema postaciami jeśli jedna z postaci jest zarażona, mierzy się czas rozpoczęcia i zakończenia kontaktu. Jeżeli ten czas spełnia warunki minimalnego czasu zarażenia, oblicza się szansę na zarażenie zgodnie z zarażalnością wirusa. W przypadku braku narażenia status pozostaje bez zmian. W sytuacji narażenia, status zmienia się na EXPOSED, a następnie oblicza się szansę na rozwinięcie choroby. Jeśli choroba się nie rozwija, status zostaje zmieniony na "healthy". W przypadku rozwinięcia choroby, status zmienia się na INFECTED. W obydwu przypadkach zmiana statusu ma miejsce po określonym czasie inkubacji (± 24h). Zarażona osoba zostaje usunięta z symulacji po wykryciu (± 24h). Algorytm przedstawiony jest na diagramie sekwencyjnym 5.1. Implementację algorytmu przedstawiają fragmenty kodu 5.2 i 5.3, dotyczą one zachowania

przy rozpoczęciu i zakończeniu kontaktu, przyglądając się funkcji *OnTriggerExit()* możemy zauważyć że to ona jest odpowiedzialna za zmianę statusu osobnika na EXPOSED oraz wywołanie kolejnej funkcji odpowiedzialnej za wyliczenie szansy na rozwinięcie się choroby 5.4.

5.3.2 Symulacja zachowań ludzkich

Ten algorytm odpowiada za symulację codziennych działań ludzi w biurze. Losuje on różne akcje, takie jak *spacerowanie, lunch, przerwa* czy *praca*. W przypadku akcji *praca, lunch* lub *przerwa*, algorytm znajduje dostępne miejsce siedzące i zajmuje je na losowy czas od 1 do 4 godzin, po czym zmienia akcję na kolejną. Jeżeli nie ma wolnego miejsca, akcja zostaje natychmiastowo zmieniona na losowe spacerowanie. W trakcie tych działań, w przypadku spotkania dwóch postaci, istnieje 1% szansa na rozpoczęcie rozmowy. Po zakończonej rozmowie losowana jest kolejna akcja. Algorytm przedstawiony jest na diagramie sekwencyjnym 5.5. Implementację pokazuje fragment kodu 5.6, zawiera dwie funkcje kontrolujące rozpoczęcie i zakończenie akcji. Do symulacji konwersacji miedzy agentami wykorzystano analogiczne bardzo podobne rozwiązanie, rozszerzone o dodatkowe kroki konieczne dla tego zachowania.



Rysunek 5.1: Sekwencyjny diagram algorytmu kontaktu miedzy agentami

```
1  private void OnTriggerEnter2D(Collider2D collider)
2  {
3      humanScript otherHuman = collider.GetComponent<humanScript>()
           ;
4      int conversation = UnityEngine.Random.Range(0,100);
5      if (conversation == 1)
6      {
7          otherHuman.Conversation();
8          Conversation();
9      }
10     if(otherHuman.GetStatus() == Status.INFECTED && status==
           Status.HEALTHY)
11     {
12         ShowRange();
13         timeEnter = Time.time;
14     }
15     if(status == Status.INFECTED && otherHuman.GetStatus() ==
           Status.HEALTHY)
16     {
17         ShowRange();
18     }
19 }
```

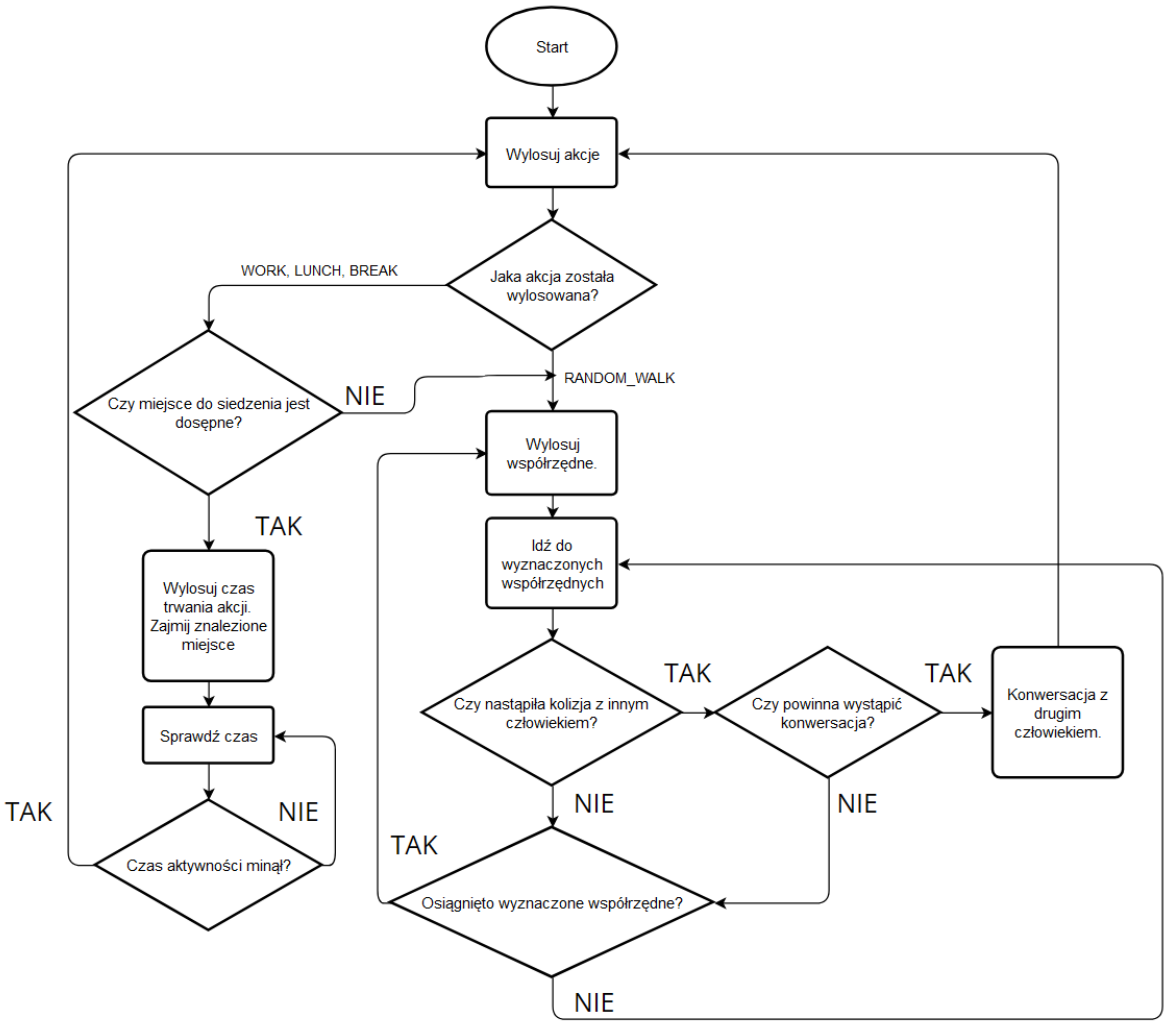
Rysunek 5.2: Funkcja obsługująca początek kontaktu miedzy agentami.

```
1       private void OnTriggerExit2D(Collider2D collider)
2       {
3           humanScript otherHuman = collider.GetComponent<
                humanScript>();
4           if(otherHuman.GetStatus() == Status.INFECTED && status
                ==Status.HEALTHY)
5           {
6               timeExit = Time.time;
7
8               float contactDuration = timeExit - timeEnter;
9
10              if (contactDuration > timeToInfection)
11              {
12                  int exposed = UnityEngine.Random.Range(0,100);
13                  if(exposed < virusSpreadFactor)
14                  {
15                      status = Status.EXPOSED;
16                      body.color = Color.yellow;
17                      simInterface.IncreaseExposed();
18                      CalculateInfection();
19                  }
20              }
21          }
22          HideRange();
23      }
```

Rysunek 5.3: Funkcja obsługująca koniec kontaktu miedzy agentami.

```
1       privatevoid CalculateInfection()
2       {
3           int infectionProbabilty = (int)(virusSpreadFactor*(1 -
                immunity));
4           int attempt = UnityEngine.Random.Range(1,100);
5           if(attempt < infectionProbabilty)
6           {
7               infectionTime = clock.GetHoursPassed() + UnityEngine
                    .Random.Range(incubationPeriod-24,
                    incubationPeriod+24);
8               willBeInfected = true;
9           }
10          else
11          {
12              infectionTime = clock.GetHoursPassed() + 24;
13              willBeInfected = false;
14          }
15      }
```

Rysunek 5.4: Funkcja wyliczająca szansę na rozwinięcie się choroby.



Rysunek 5.5: Sekwencyjny diagram algorytmu symulacji zachowań ludzkich

```
1       private void StartActivity()
2       {
3           activity = (Activity)UnityEngine.Random.Range(0,4);
4           bool seatFound = false;
5           switch(activity)
6           {
7               case Activity.WORK:
8               seatFound = SearchSeat(desks);
9               break;
10              case Activity.BREAK:
11              seatFound = SearchSeat(sofas);
12              break;
13              case Activity.LUNCH:
14              seatFound = SearchSeat(kitchenSeats);
15              break;
16              default:
17              break;
18          }
19          if(!seatFound)
20          {
21              activity = Activity.RANDOM_WALK;
22          }
23      }
24
25      private void EndActivity()//wywolywana w funkcji Update()
26      {
27          if(currentActivityEndTime < clock.GetHour())
28          {
29              currentSeat.SetOccupation(false);
30              StartActivity();
31          }
32      }
```

Rysunek 5.6: Funkcja wyliczająca szansę na rozwinięcie się choroby.

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

W tym rozdziale przedstawione zostaną techniki sprawdzania i potwierdzania poprawności działania aplikacji. W procesie tworzenia oprogramowania istotne jest nie tylko zaplanowanie i implementacja funkcji, ale także dokładne sprawdzenie, czy rezultaty są zgodne z założeniami projektowymi. Weryfikacja koncentruje się na sprawdzeniu, czy projekt spełnia założenia funkcjonalne i techniczne, natomiast walidacja ocenia, czy to, co zostało zaimplementowane, odpowiada rzeczywistym potrzebom użytkowników. W dalszej części tego rozdziału przedstawione zostaną metody, narzędzia i procedury stosowane w procesie weryfikacji i walidacji.

6.1 Sposoby testowania i organizacja eksperymentów

W ramach przeprowadzonych testów aplikacji dokonano ręcznego dostosowania parametrów symulacji. Obejmowały one zakres od maksymalnych do minimalnych wartości oraz różne kombinacje pomiędzy nimi. Kluczowe parametry zostały testowane oddzielnie, a przykładowo długość symulacji poddana była analizie, aby ocenić wpływ zwiększenia liczby cykli dnia i nocy na funkcjonalność. Dodatkowo przetestowano różne wartości procentowe dla zarażonych i osób noszących maseczki. Eksperymentowano również z różnymi liczebnościami populacji, aby zrozumieć, jak wpływają one na symulację zachowań ludzkich.

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] David Adam. „Special report: The simulations driving the world's response to COVID-19". W: *Nature* 580.7802 (2020). Gale Academic OneFile, s. 316. URL: https://link.gale.com/apps/doc/A619849970/AONE?u=anon~b61079c3&sid=googleScholar&xid=ea3f7606.

[2] 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.

[3] Dirk Brockmann. „Global Connectivity and the Spread of Infectious Diseases". W: *Nova Acta Leopoldina*. 419. Robert Koch-Institut, 2017. DOI: http://dx.doi.org/10.25646/2797.

[4] Y-S Tsai C-Y Huang i T-H Wen. „Simulations for epidemiology and public health education". W: *Journal of Simulation* 4.1 (2010), s. 68–80. DOI: 10.1057/jos.2009.13. eprint: https://doi.org/10.1057/jos.2009.13. URL: https://doi.org/10.1057/jos.2009.13.

[5] CDC. *The Discovery and Reconstruction of the 1918 Pandemic Virus*. Centers for Disease Control and Prevention. 2019.

[6] Kathy Kotiadis Thomas Monks Bhakti Stephan Onggo Duncan A. Robertson Christine S.M. Currie John W. Fowler i Antuela A. Tako. „How simulation modelling can help reduce the impact of COVID-19". W: *Journal of Simulation* 14.2 (2020), s. 83–97. DOI: 10.1080/17477778.2020.1751570. eprint: https://doi.org/10.1080/17477778.2020.1751570. URL: https://doi.org/10.1080/17477778.2020.1751570.

[7] Ndemic Creations. *Plague Inc.* URL: https://play.google.com/store/apps/details?id=com.miniclip.plagueinc (term. wiz. 30. 12. 2023).

[8] 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.

[9] 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.

[10] Mohammad Amin Khazeei Tabari, Hooman Khoshhal, Alireza Tafazoli, Mohanna Khandan i Abouzar Bagheri. „Applying computer simulations in battling with COVID-19, using pre-analyzed molecular and chemical data to face the pandemic". W: *Informatics in Medicine Unlocked* 21 (2020), s. 100458. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2020.100458. URL: https://www.sciencedirect.com/science/article/pii/S2352914820306080.

[11] Adrian Kopec. *Epidemic Simulator*. URL: https://play.google.com/store/apps/details?id=com.kopecsec.EpidemicSimulator&hl=pl&gl=US (term. wiz. 30. 12. 2023).

[12] Imię Nazwisko i Imię Nazwisko. *Modeling Disease Spread*. URL: https://www.biointeractive.org/classroom-resources/modeling-disease-spread (term. wiz. 30. 12. 2023).

[13] Hilda P. Hudson Ronald Ross. „An application of the theory of probabilities to the study of a priori pathometry.—Part II". W: *Proc. R. Soc. Lond. A* 93 (maj 1917), s. 212–225. DOI: https://doi.org/10.1098/rspa.1917.0014.

[14] Hilda P. Hudson Ronald Ross. „An application of the theory of probabilities to the study of a priori pathometry.—Part III". W: *Proc. R. Soc. Lond. A* 93 (maj 1917), s. 225–240. DOI: https://doi.org/10.1098/rspa.1917.0015.

[15] Ronald Ross. „An application of the theory of probabilities to the study of a priori pathometry.—Part I". W: *Proc. R. Soc. Lond. A* 92 (lut. 1916), s. 204–230. DOI: https://doi.org/10.1098/rspa.1916.0007.

[16] 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.

[17] A. G. McKendrick William Ogilvy Kermack. „An application of the theory of probabilities to the study of a priori pathometry.—Part III". W: *Proc. R. Soc. Lond. A* 115 (sierp. 1927), s. 700–721. DOI: https://doi.org/10.1098/rspa.1927.0118.

Dodatki

Spis skrótów i symboli

SIR	Suceptible, Infected, Removed (podatni-zainfekowani-ozdrowieńcy)
SEIR	Suceptible, Exposed, Infected, Removed (podatni-wystawieni-zainfekowani-ozdrowieńcy)
P_z	szansa na zarażenie
R_0	współczynnik zakażalności patogenu
I	odporność agenta na zarażenie.
$T_{inkubacji}$	średni czas inkubacji patogenu.
$T_{wykrycia}$	średni czas wykrycia choroby i odizolowania agenta.
InfektoSym	nazwa aplikacji wykonanej w ramach projektu.

Źródła

```
1       private void OnTriggerEnter2D(Collider2D collider)
2       {
3           humanScript otherHuman = collider.GetComponent<
                humanScript>();
4           int conversation = UnityEngine.Random.Range(0,100);
5           if (conversation == 1)
6           {
7               otherHuman.Conversation();
8               Conversation();
9           }
10          if(otherHuman.GetStatus() == Status.INFECTED && status
                ==Status.HEALTHY)
11          {
12              ShowRange();
13              timeEnter = Time.time;
14          }
15          if(status == Status.INFECTED && otherHuman.GetStatus()
                == Status.HEALTHY)
16          {
17              ShowRange();
18          }
19      }
```

Rysunek 1: Funkcja obsługująca początek kontaktu miedzy agentami.

```
1       private void OnTriggerExit2D(Collider2D collider)
2       {
3           humanScript otherHuman = collider.GetComponent<
                humanScript>();
4           if(otherHuman.GetStatus() == Status.INFECTED && status
                ==Status.HEALTHY)
5           {
6               timeExit = Time.time;
7
8               float contactDuration = timeExit - timeEnter;
9
10              if (contactDuration > timeToInfection)
11              {
12                  int exposed = UnityEngine.Random.Range(0,100);
13                  if(exposed < virusSpreadFactor)
14                  {
15                      status = Status.EXPOSED;
16                      body.color = Color.yellow;
17                      simInterface.IncreaseExposed();
18                      CalculateInfection();
19                  }
20              }
21          }
22          HideRange();
23      }
```

Rysunek 2: Funkcja obsługująca koniec kontaktu miedzy agentami.

```
1       privatevoid CalculateInfection()
2       {
3           int infectionProbabilty = (int)(virusSpreadFactor*(1 -
                immunity));
4           int attempt = UnityEngine.Random.Range(1,100);
5           if(attempt < infectionProbabilty)
6           {
7               infectionTime = clock.GetHoursPassed() + UnityEngine
                    .Random.Range(incubationPeriod-24,
                    incubationPeriod+24);
8               willBeInfected = true;
9           }
10          else
11          {
12              infectionTime = clock.GetHoursPassed() + 24;
13              willBeInfected = false;
14          }
15      }
```

Rysunek 3: Funkcja wyliczająca szansę na rozwinięcie się choroby.

```
1       private void StartActivity()
2       {
3           activity = (Activity)UnityEngine.Random.Range(0,4);
4           bool seatFound = false;
5           switch(activity)
6           {
7               case Activity.WORK:
8               seatFound = SearchSeat(desks);
9               break;
10              case Activity.BREAK:
11              seatFound = SearchSeat(sofas);
12              break;
13              case Activity.LUNCH:
14              seatFound = SearchSeat(kitchenSeats);
15              break;
16              default:
17              break;
18          }
19          if(!seatFound)
20          {
21              activity = Activity.RANDOM_WALK;
22          }
23      }
24
25      private void EndActivity()//wywolywana w funkcji Update()
26      {
27          if(currentActivityEndTime < clock.GetHour())
28          {
29              currentSeat.SetOccupation(false);
30              StartActivity();
31          }
32      }
```

Rysunek 4: Funkcja wyliczająca szansę na rozwinięcie się choroby.

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