



**Politechnika
Śląska**

PROJEKT INŻYNIERSKI

Aplikacja do symulacji rozprzestrzeniania się zarażeń

Jakub CIOŁEK

Nr albumu: 295618

Kierunek: Informatyka

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

Na podstawie uznanych modeli rozprzestrzeniania się zarażeń oraz analizy istniejącego oprogramowania badającego epidemie, opracowano aplikację symulacyjną, której głównym celem jest ukazanie w czasie rzeczywistym sposobu rozprzestrzeniania się chorób w środowisku biurowym. Aplikacja łączy zmodyfikowany matematyczny model symulacji z podejściem agentowym, co zapewnia bardziej precyzyjne odwzorowanie rzeczywistości. Szczególną uwagę skupiono na dostarczeniu prostego i zrozumiałego interfejsu wizualnego, umożliwiającego zrozumienie mechanizmów stojących za zjawiskami epidemiologicznymi nawet osobom nieposiadającym specjalistycznej wiedzy medycznej.

Słowa kluczowe

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

Thesis title

Application for simulating the spread of infections.

Abstract

Based on recognized models of infection spread and an analysis of existing epidemic research software, a simulation application has been developed. Its primary goal is to showcase, in real-time, the way diseases spread in a office environment. The application combines a modified mathematical simulation model with an agent-based approach, providing a more precise representation of reality. Special attention has been given to delivering a simple and understandable visual interface, allowing individuals without specialized medical knowledge to comprehend the mechanisms behind epidemiological phenomena.

Key words

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

Spis treści

Wstęp	2
1 Modele symulacji rozprzestrzeniania się zarażeń	3
1.1 Modele bazujące na SIR	4
1.2 Modele agentowe	5
1.3 Inne aplikacje do symulacji zarażeń	7
1.3.1 EpiSimdemics - równoległy algorytm symulacji epidemii . .	7
1.3.2 Zastosowanie rozszerzonej rzeczywistości w symulacji zarażeń	8
1.3.3 Aplikacje dostępne w internecie i sklepach z aplikacjami . .	8
2 Wymagania i narzędzia	11
2.1 Wymagania funkcjonalne i нефункционалне	11
2.1.1 Wymagania funkcjonalne	11
2.1.2 Wymagania нефункционалне	13
2.2 Wybrana technologia	14
2.3 Model symulacji rozprzestrzeniania się zarażeń zastosowany w InfektoSym	14
2.3.1 Równania opisujące model symulacji	15
2.3.2 Symulacja zachowań ludzkich	16
3 Specyfikacja zewnętrzna	19
3.1 Wymagania	19
3.2 Uruchamianie aplikacji	19
3.3 Obsługa aplikacji	20
4 Specyfikacja wewnętrzna	25
4.1 Architektura systemu	25
4.1.1 Biblioteki i moduły	26
4.2 Algorytmy	26
4.2.1 Kontrola rozprzestrzeniania się choroby	27
4.2.2 Symulacja zachowań ludzkich	27

5 Weryfikacja i walidacja	31
5.1 Sposoby testowania i organizacja eksperymentów	31
5.2 Przypadki testowe	31
5.3 Wykryte i usunięte błędy	32
Podsumowanie i wnioski	36
Bibliografia	38
Spis skrótów i symboli	41
Źródła	43
Lista dodatkowych plików, uzupełniających tekst pracy	47
Spis rysunków	49
Spis tabel	51

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 stworzyła 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 [5].

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 [11][1][7]. 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 stwarza sposobność do skuteczniejszego przygotowania się do potencjalnych zagrożeń zdrowotnych [4].

Celem niniejszej pracy jest stworzenie aplikacji, dedykowanej symulacji rozprzestrzeniania się epidemii, ze szczególnym uwzględnieniem środowiska biurowego. Głównym założeniem projektu jest dostarczenie narzędzia umożliwiającego szeroką parametryzację symulacji, co pozwoli na elastyczne dostosowanie scenariuszy do różnych warunków i kontekstów. Aplikacja ma stanowić realistyczny model symulacyjny, uwzględniający codzienne zachowania ludzkie w biurze, a także umożliwiać interaktywną wizualizację procesu rozprzestrzeniania się patogenów w czasie rzeczywistym.

Pracę podzielono na siedem rozdziałów. Na początku pracy dokonano analizy istniejących 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 1

Modele symulacji rozprzestrzeniania się zarażeń

Symulacje komputerowe są niezwykle istotnym narzędziem badawczym, umożliwiającym testowanie scenariuszy epidemiologicznych co pozwala na przygotowanie się do różnych sytuacji kryzysowych. Przyczynia się to 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, w pierwszej kolejności konieczne jest zrozumienie mechanizmów, które kierują postępującą epidemią. 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 patogen. 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.

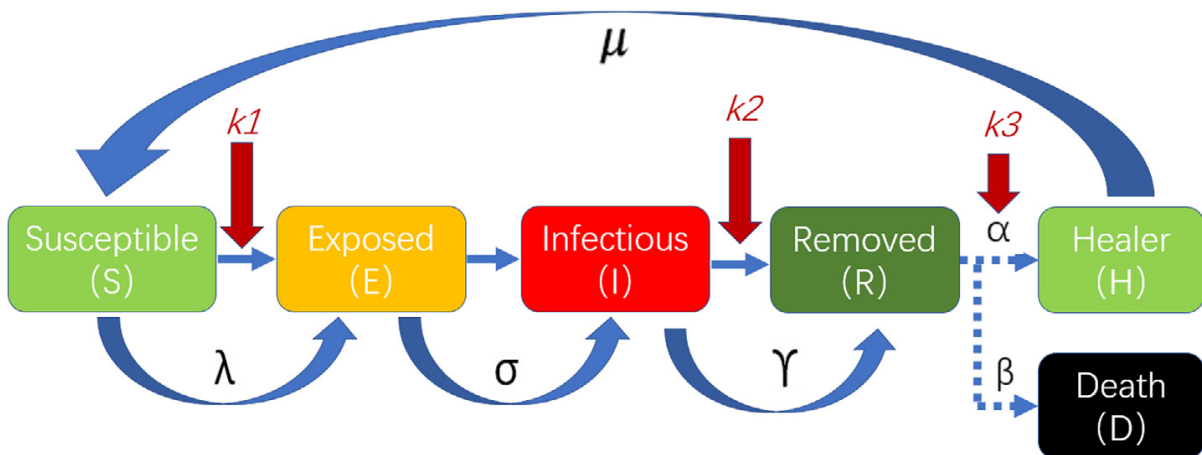
Wzmózione prace w dziedzinie modelowania epidemiologicznego miały miejsce na początku XX wieku, prowadzone przez takich badaczy jak Ronald Ross[15] i Hilda Phoebe Hudson[13][14]. Świat po I wojnie światowej stanął przed pandemią grypy hiszpanki, która zarażając 1/3 ówczesnej populacji i powodując więcej ofiar niż globalny konflikt zbrojny[6],

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 (Susceptible-Infected-Removed). 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.

1.1 Modele bazujące na SIR

We współczesnych badaniach często rozwija się model SIR, tak aby mógł 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 [9].

We wspomnianym modelu rozszerzono oryginalny SIR o dodatkową grupę E - *Exposed* (narażeni) oraz czynnik K , który określa działania przeciwdziałające epidemii 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 1.1.



Rysunek 1.1: Schemat działania modelu K -SEIR[9]

Badacze przenieśli teoretyczny model K -SEIR do praktyki, tworząc intuicyjne oprogramowanie w języku PYTHON. Szczególną uwagę zwrócili na projektowanie interfejsu graficznego użytkownika, logikę sterowania, operacje matematyczne, precyzję obliczeń, zarządzanie prędkością symulacji, czytelność wizualizacji danych, a także możliwość importu i eksportu danych. Ich działania dostarczyły narzędzie, które nie tylko efektywnie realizuje teoretyczny model, ale również jest przyjazne dla użytkownika, oferując intuicyjny interfejs, dostosowany do potrzeb użytkowników.

Tabela 1.1: Opis modelu epidemiologicznego K -SEIR opracowany na podstawie[9].

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
	$\alpha + \beta = \gamma$	
	$s_0 + e_0 + i_0 + r_0 = N$ (dla $t = 0$)	0: czas $t = 0$
	$\lambda_k = (1 - k_1)\lambda$	K : współczynnik interwencji ludzkiej
	$\gamma_k = k_2\gamma$ $\alpha_k = k_3\alpha$	k_1 : miara izolacji fizycznej, współczynnik λ k_2 : zdolność przyjęcia do szpitala, współczynnik γ k_3 : zdolność leczenia, współczynnik α

1.2 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, np. 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.

W artykule [10], poruszającym temat modelowania agentowego, możemy przeczytać szczegółowe informacje na temat tego co badacze wzięli pod uwagę podczas tworzenia swojego modelu. Był on oparty o dane populacji, umiejscowienia szkół i miejsc pracy a także informacje dotyczące szczepień. Rzeczywiste dane są używane do określenia struktury wiekowej i płciowej populacji wraz z właściwym rozkładem wielkości gospodarstw domowych oraz innymi cechami takimi jak wiek i ilość dzieci. Następnie dane opisujące lokalizację szkół oraz miejsc pracy dają możliwość wiernego odwzorowania interakcji pomiędzy ludźmi. W celu oceny podatności populacji na rozprzestrzenianie się choroby uwzględniono ilość szczepień (model skupiony był 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 rozmieszczono agentów w odpowiednich domach według danych o populacji. Na koniec uwzględniono dodatkowe czynniki między innymi transport. Ostatnim krokiem było matematyczne opisanie szans na zarażeniu, 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:

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

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 przykładzie wielu miastach w Irlandii, ze względu na dostępne dane co do ich struktury oraz populacji. Dodatkowo model został porównany z rzeczywistymi danymi 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 epidemii, w którym więcej niż 30 agentów zostało zainfekowanych. Wyniki pokazują, że chociaż średnia dla wszystkich prób jest niższa niż liczba zainfekowanych w przypadku epidemii w Schull, to liczba faktycznie zainfekowanych osób znajduje się w 75. centylu wyników modelu.” (...) [10]

Podsumowując, przewagą modeli agentowych nad modelami takimi jak SIR 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 taki model traci na znaczeniu w miarę zwiększania się populacji, dlatego idealnie nadaje się do symulacji epidemii w małych miastach ale nie w obrębie całego kraju lub świata.

1.3 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.

1.3.1 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 przechwytyjącego 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 danych na temat rozprzestrzeniania się choroby w populacji, obejmujących takie detale jak: konkretny zestaw osób zainfekowanych, miejsce zarażenia oraz informacji kto ich zarażał.

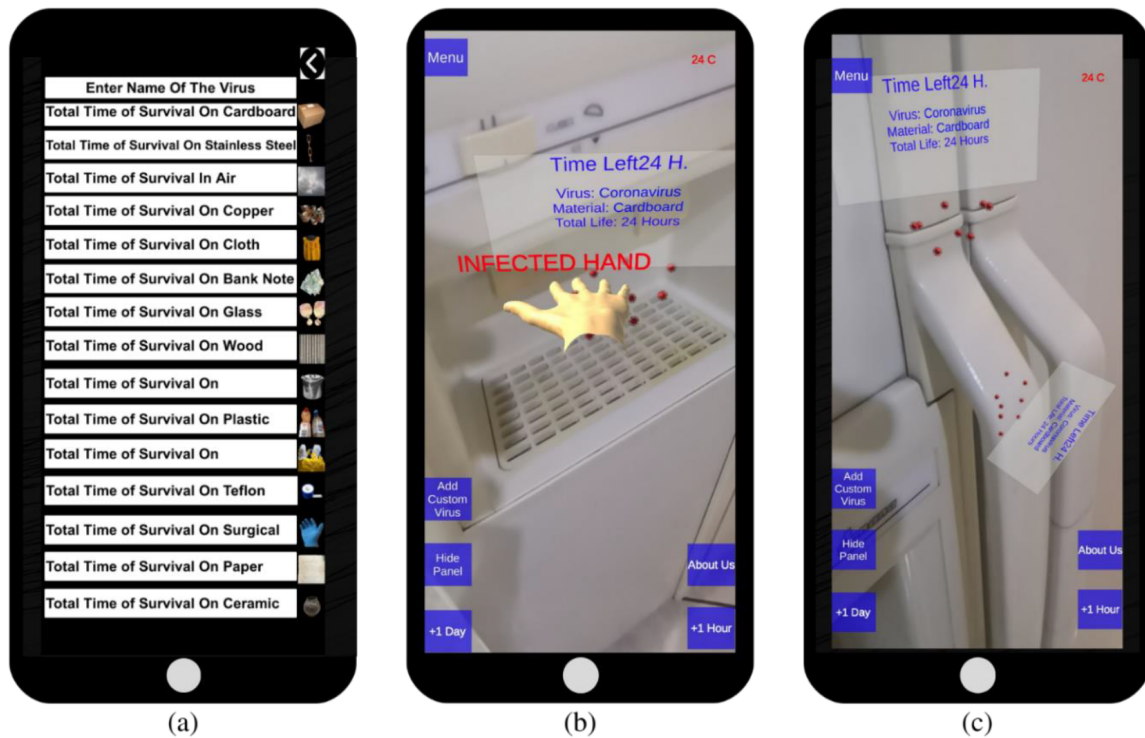
1.3.2 Zastosowanie rozszerzonej rzeczywistości w symulacji zarażeń

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 1.2. 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]

1.3.3 Aplikacje dostępne w internecie i sklepach z aplikacjami

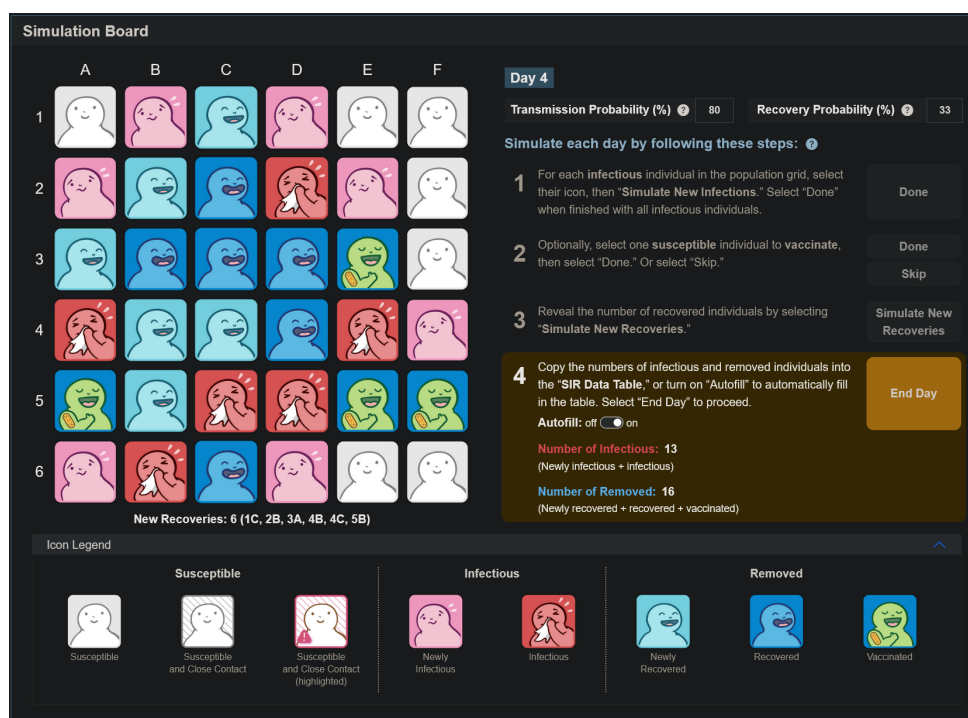
W dzisiejszym świecie aplikacji internetowych oraz sklepów z aplikacjami można znaleźć różnorodne narzędzia związane z modelowaniem i symulacją zarażeń. Jednym z przykładów jest aplikacja stworzona przez Biointeractive „*Modeling Disease Spread*” [3], która umożliwia interaktywne śledzenie modelu SIR. Warto jednak zauważyć, że ta konkretna aplikacja, skoncentrowana głównie na prezentacji wykresów i danych, jest raczej statycznym narzędziem 1.3.

Przejsie 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

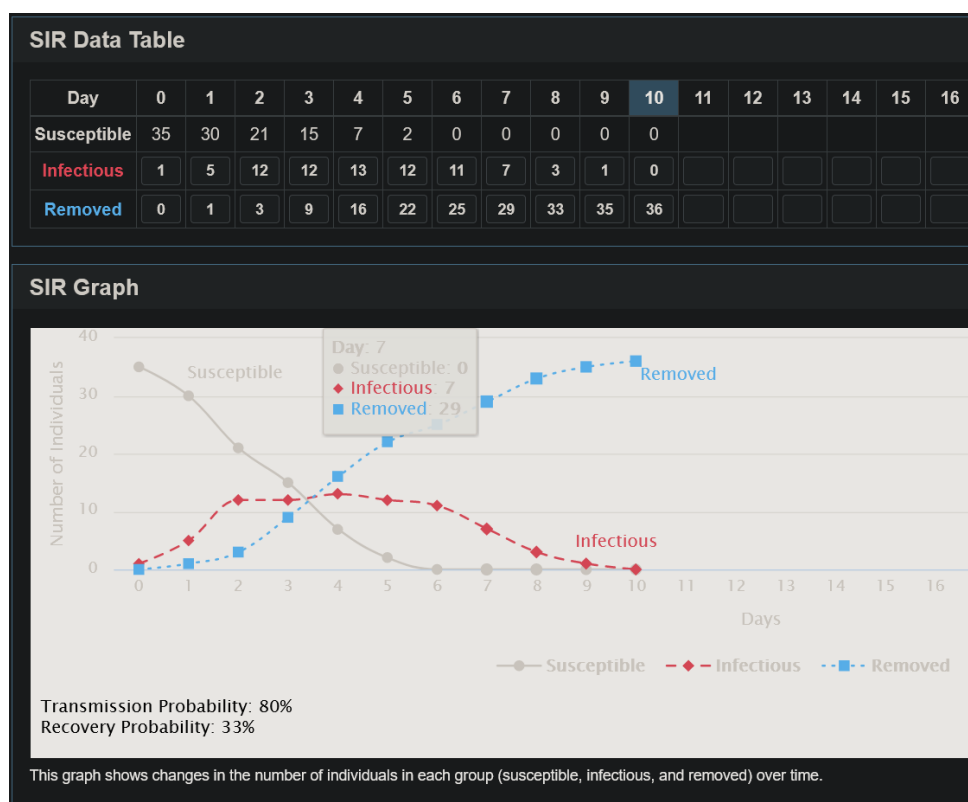


Rysunek 1.2: Demonstracja działania aplikacji: (a) dostosowywanie niestandardowego wirusa, (b) wprowadzenie wirusa w rzeczywistego świata, (c) wirus widoczny w rozszerzonej rzeczywistości z panelem informacyjnym [16].

naukowego czy edukacyjnego narzędzia [8] [12]. 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 [3].



(b) Statystyki generowane przez aplikację [3].

Rysunek 1.3: Wygląd aplikacji „Modeling Disease Spread” [3].

Rozdział 2

Wymagania i narzędzia

Rozdział omawia kluczowe aspekty dotyczące aplikacji do symulowania rozprzestrzeniania się zarażeń o nazwie *InfektoSym*. Rozpocznie się od analizy i przedstawienia wymagań funkcjonalnych oraz нефункциональных. 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.

2.1 Wymagania funkcjonalne i нефункциональные

Skoncentrowano się na opracowaniu symulacji, ze szczególnym naciskiem na środowisko wewnątrz budynków. Taka decyzja wynika z faktu, że ludzie spędzają większość swojego czasu w budynkach, a duże zagęszczenie populacji w takich miejscach stwarza optymalne warunki dla rozprzestrzeniania się chorób.

2.1.1 Wymagania funkcjonalne

- **Symulacja ruchu i interakcji:**

Aplikacja pozwala na śledzenie 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 środki ochrony osobistej 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 procentowej naturalnej odporności 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.

2.1.2 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 odporna 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.

2.2 Wybrana technologia

Rozważając wybór technologii do stworzenia aplikacji *InfektoSym*, zdecydowano się na silnik gier Unity. Wybór ten 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.

2.3 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.

2.3.1 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 patogenem 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:

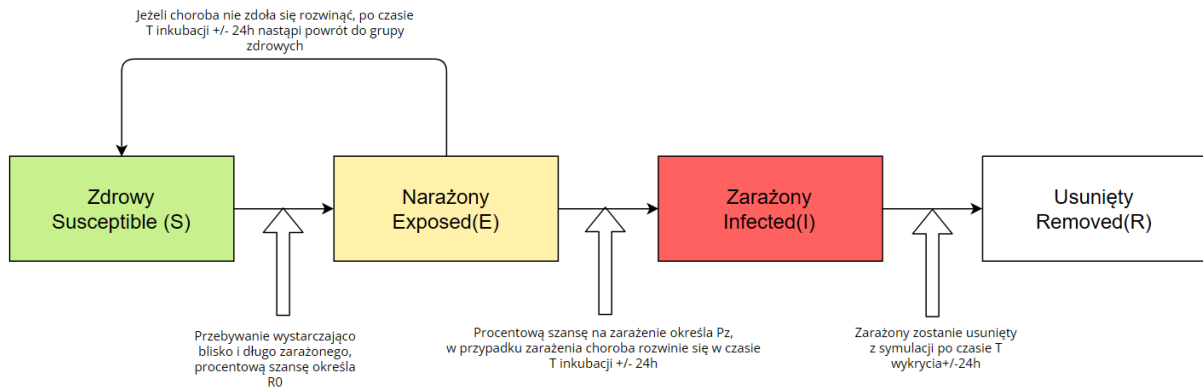
$P_z \in < 0, 100 >$ - szansa na zostanie zarażonym, wyrażona w %

$R_0 \in < 0, 100 >$ - współczynnik zakaźności 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 2.1: Schemat działania modelu użytego w *InfektoSym*

2.3.2 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:**
 - 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 biurka, 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 3

Specyfikacja zewnętrzna

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.

3.1 Wymagania

Sprzęt i oprogramowanie potrzebne do uruchomienia aplikacji:

- Komputer z systemem operacyjnym Windows.
- Ekran o rozdzielczości co najmniej 1280x720 pikseli.
- Karta graficzna wspierająca OpenGL 3.2 lub nowszy.
- System operacyjny: Windows 7/8/10 i nowsze.
- Zainstalowany runtime Unity w wersji zgodnej z aplikacją.

3.2 Uruchamianie aplikacji

Aby skorzystać z aplikacji, należy wykonać poniższe kroki:

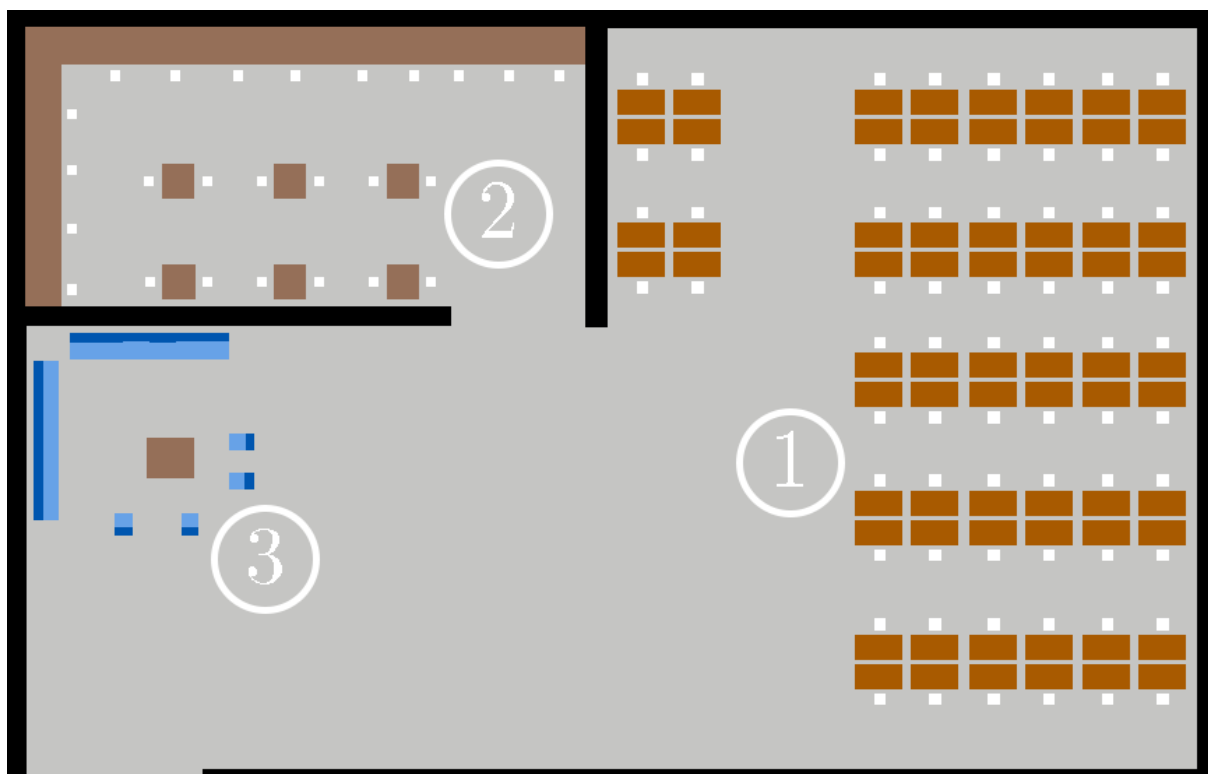
1. Pobrać instalator aplikacji.
2. Uruchomić instalator.
3. Postępować zgodnie z instrukcjami aż do zakończenia instalacji.
4. Uruchomić aplikację dwukrotnie klikając jej skrót lub plik wykonwalny *exe* znajdujący się w folderze instalacji.

3.3 Obsługa aplikacji

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

- **Wizualizacja**

Graficzne przedstawienie symulacji znajduje się po lewej stronie okna aplikacji. Mapa biura, odzwierciedla przestrzeń do pracy, kuchnię oraz miejsce wypoczynku. Szczegółowy opis mapy i jej poszczególnych elementów znajduje się na rysunku 3.1.



Rysunek 3.1: Mapa biura: 1. Biurka do pracy, 2. Kuchnia ze stolikami, 3. Miejsce wypoczynku

- **Ustawianie Parametrów Symulacji**

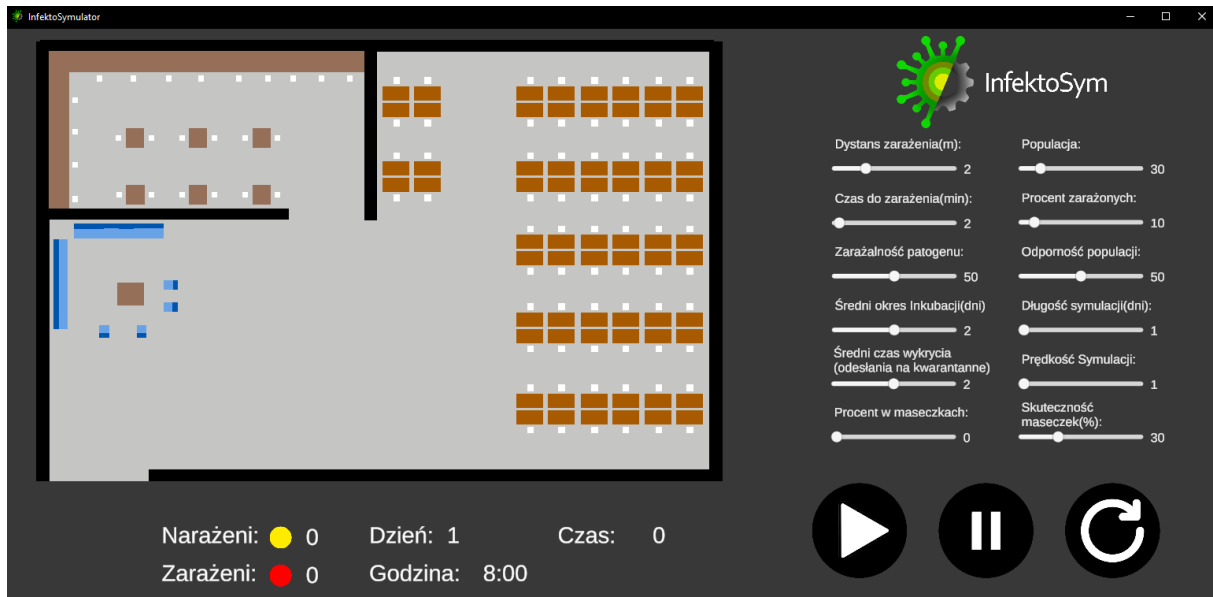
Dostosowywanie parametrów odbywa się poprzez ustawianie suwaków znajdujących się z prawej strony (widocznych na rysunku 3.3 przy numerze pierwszym) za pomocą myszki. Parametry należy ustawić przed rozpoczęciem symulacji, ich zmiana w jej trakcie nie wpłynie na jej przebieg, a nowe ustawienia zostaną wprowadzone dopiero po restarcie symulacji. Wyjątkiem jest prędkość symulacji, którą można zmieniać w trakcie działania programu.

Lista dostępnych parametrów:

- **Dystans zarażenia:**

Dystans przenoszenia się wirusa.

- **Czas do zarażenia:**
Czas(minuty) bliskiego kontaktu z chorym, aby doszło do zarażenia.
- **Zaraźliwość patogenu:**
Zdolność wirusa do zarażania.
- **Średni okres inkubacji:**
Czas(dni) od narażenia do rozwinięcia się objawów.
- **Procent populacji noszący maseczki:**
Procentowa ilość ludzi noszących maseczki.
- **Skuteczność maseczek:**
Określa jak bardzo skuteczne są maseczki w ograniczaniu zasięgu wirusa.
- **Liczebność populacji:**
Ogólna liczba postaci uczestniczących w symulacji.
- **Początkowy procent zarażonych:**
Procentowa liczba zarażonych na początku symulacji.
- **Odporność populacji:**
Odporności osobników na zarażenie.
- **Długość symulacji:**
Ilość dni trwania symulacji.
- **Prędkość symulacji:**
Umożliwienie regulacji prędkości symulacji, włączając przyspieszenie do 100-krotności normalnej prędkości.
- **Średni czas wykrycia:**
Czas(dni) jaki upływa od momentu pojawienia się objawów do ich wykrycia i zlecenia kwarantanny.
- **Rozpoczęcie symulacji i jej kontrola**
Do kontroli symulacji służą przyciski widoczne na rysunku 3.3 przy numerze drugim. Są to *Start*, *Pauza* i *Restart*.
 - **Start**
Uruchamia symulację lub ją wznowia.
 - **Pauza**
Zatrzymuje symulację. Po jej wznowieniu parametry symulacji, pozycje agentów i czas się nie zmieniają.
 - **Restart**
Kończy symulację. Usuwa wszystkich agentów, po uruchomieniu symulacji agenci zostaną stworzeni na nowo wraz ze zaktualizowanymi parametrami.



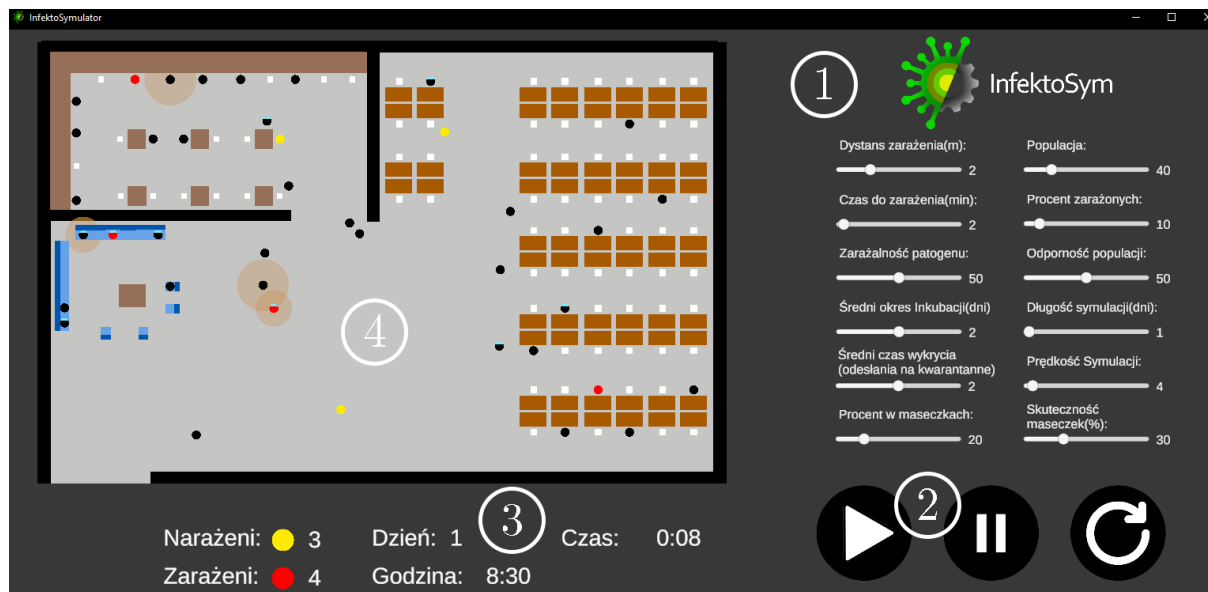
Rysunek 3.2: Ekran początkowy aplikacji.

- **Zakończenie symulacji**

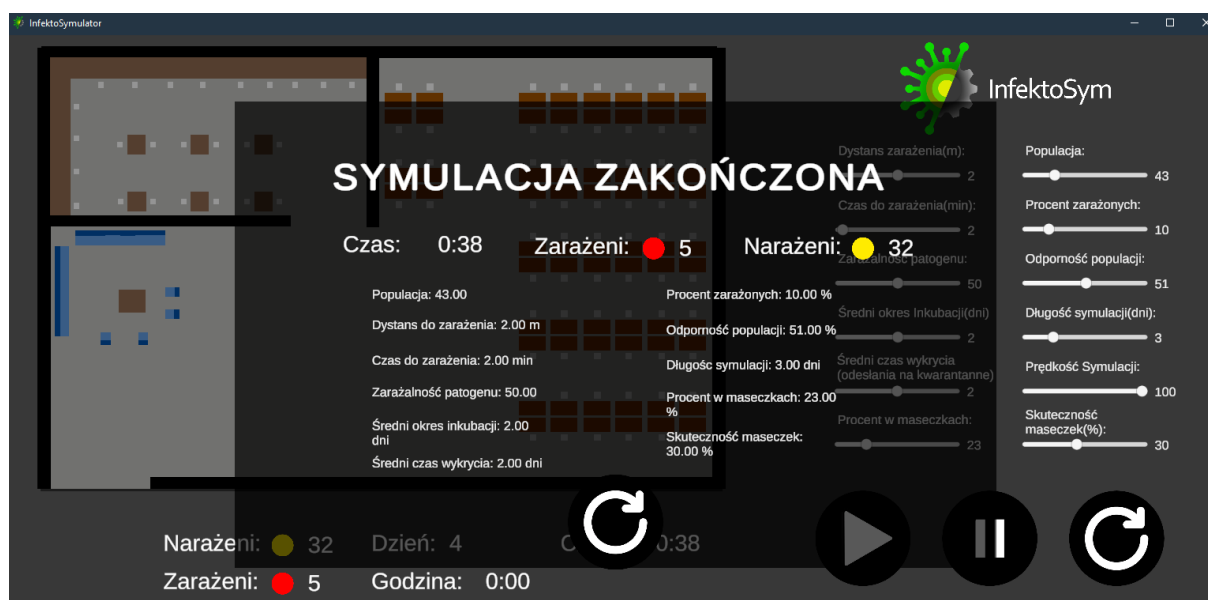
Użytkownik może zakończyć symulację w każdej chwili klikając w przycisk *Restart*. W przeciwnym przypadku symulacja zakończy się automatycznie po ustawionym na początku czasie jej trwania oraz zostanie wyświetlony ekran z podsumowaniem widoczny na rysunku 3.4.

- **Statystyki symulacji**

Podczas trwania symulacji możliwe jest bieżące monitorowanie jej statystyk, co widać na rysunku 3.2 przy numerze trzecim. Są to: liczba osób narażonych, liczba zarażonych, aktualny czas w środowisku symulacyjnym oraz rzeczywisty czas od rozpoczęcia.



Rysunek 3.3: Ekran podczas działającej symulacji: 1. Suwaki do ustawiania parametrów, 2. Przyciski kontrolujące 3. Aktualne statystyki i czas, 4. Wizualizacja



Rysunek 3.4: Ekran zakończenia symulacji

Rozdział 4

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 oraz fragmenty kodu. Celem tego rozdziału jest szczegółowe przybliżenie struktury wewnętrznej projektu, zrozumienie jego komponentów oraz logiki działania.

4.1 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 w 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.

- **GameOverScript:**

- Prosty skrypt odpowiedzialny pokazanie ekranu końcowego z odpowiednimi wynikami i parametrami symulacji.

4.1.1 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.

4.2 Algorytmy

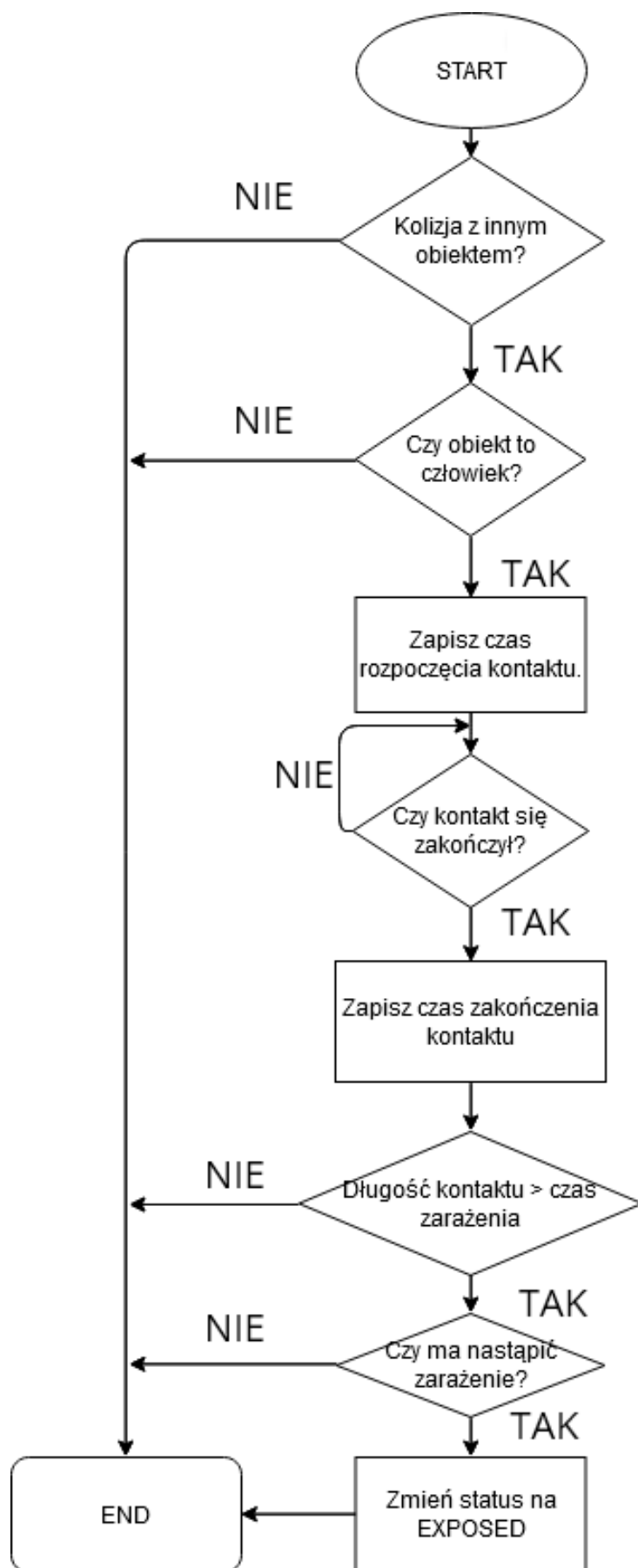
W projekcie występuje kilka algorytmów kontrolujących symulację, zajmujących się kontrolą czasu, cyklem dnia i nocy, symulacją przenoszenia się choroby oraz symulacją ludzkich zachowań. Poniżej przedstawiono opis dwóch kluczowych algorytmów.

4.2.1 Kontrola rozprzestrzeniania się choroby

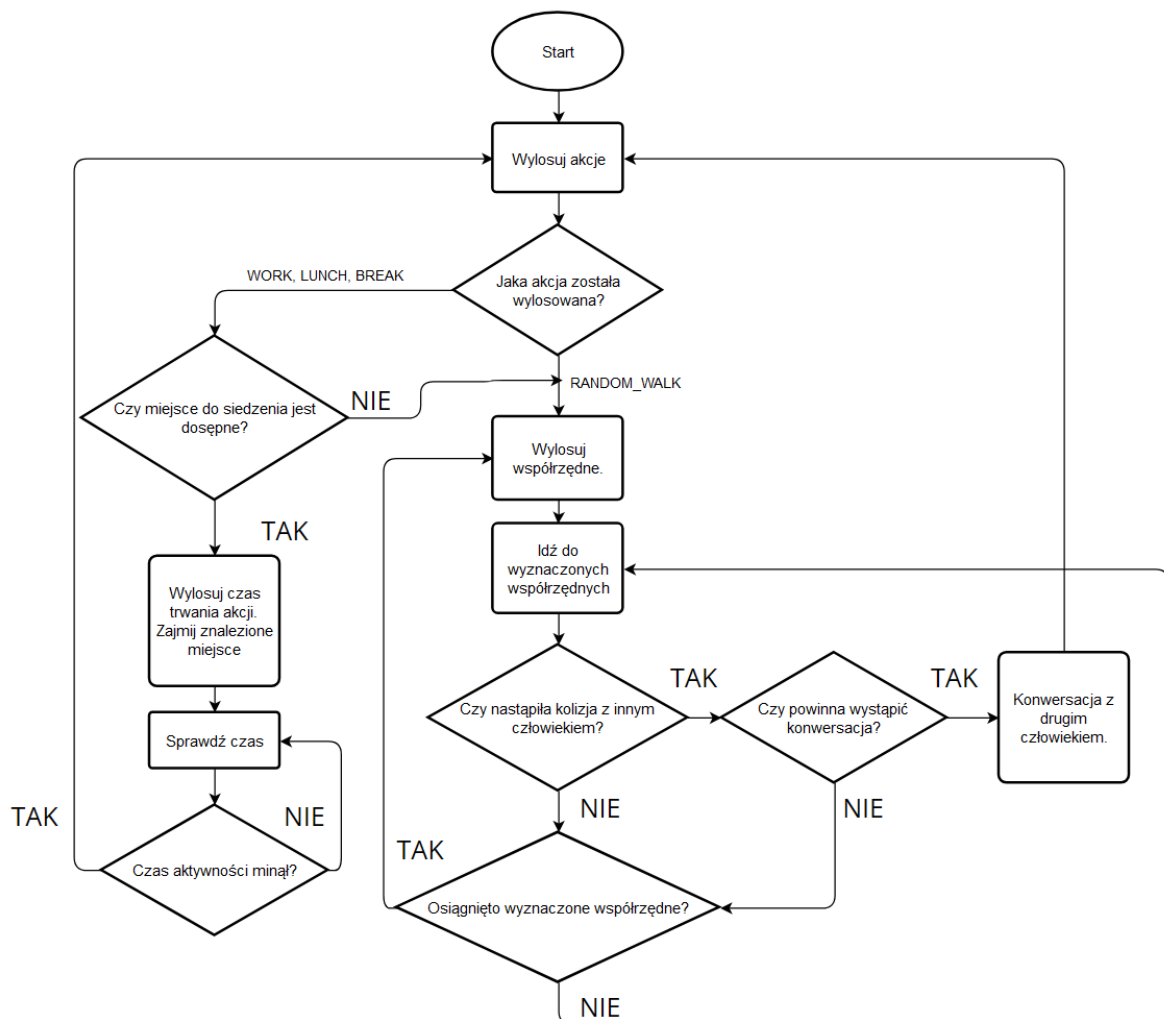
Algorytm ten skupia się na precyzyjnej symulacji interakcji między osobami. W przypadku kolizji między dwiema postaciami, jeśli jedna z postaci jest zarażona, mierzy się czas rozpoczęcia i zakończenia kontaktu. Jeżeli ten czas spełnia warunki minimalnego czasu zarażenia, oblicza się szansę na zarażenie 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 ($\pm 24h$). Zarażona osoba zostaje usunięta z symulacji po czasie wykrycia ($\pm 24h$). Algorytm przedstawiony jest na diagramie sekwencyjnym 4.1. Implementację algorytmu przedstawiają fragmenty kodu 1 i 2, 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 3.

4.2.2 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 4.2. Implementację pokazuje fragment kodu 4, zawierający dwie funkcje kontrolujące rozpoczęcie i zakończenie akcji. Do symulacji konwersacji między agentami wykorzystano analogiczne rozwiązanie, rozszerzone o dodatkowe kroki konieczne dla tego zachowania.



Rysunek 4.1: Sekwencyjny diagram algorytmu kontaktu między agentami



Rysunek 4.2: Sekwencyjny diagram algorytmu symulacji zachowań ludzkich

Rozdział 5

Weryfikacja i walidacja

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.

5.1 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.

5.2 Przypadki testowe

Tabela 5.1 przedstawia różnorodne przypadki testowe dla parametrów symulacji w aplikacji. Testy zostały zaplanowane w celu sprawdzenia zachowania symulacji w ekstremalnych warunkach oraz w warunkach typowych. Poniżej znajduje się krótki opis każdego testu:

- **Test 1 (Maksymalne wartości):** Ten test uwzględnia maksymalne wartości dla każdego z parametrów symulacji. Jest to scenariusz ekstremalny, mający na celu zidentyfikowanie, czy aplikacja poprawnie obsługuje ekstremalne warunki, takie jak bardzo zaraźliwe środowisko, długi czas inkubacji czy duża liczba populacji.
- **Test 2 (Minimalne wartości):** W tym teście wszystkie parametry przyjmują minimalne wartości. Celem jest sprawdzenie, czy aplikacja jest odporna na sytuacje, w których czynniki zakaźne są minimalne, a populacja jest ograniczona.
- **Test3 (Wartości domyślne):** Test ten obejmuje wartości domyślne dla wszystkich parametrów symulacji. Stanowi punkt odniesienia, w którym sprawdzane jest standardowe zachowanie symulacji.
- **Testy 4 - 8 (Losowe wartości):** Pozostałe testy obejmują losowo wybrane wartości parametrów symulacji. Te przypadki mają na celu ocenę ogólnej elastyczności i skuteczności aplikacji w zróżnicowanych warunkach. Wartości te zostały dobrane w celu odtworzenia różnorodnych scenariuszy, które mogą wystąpić w rzeczywistych warunkach.

Tabela 5.1: Parametry symulacji użyte w poszczególnych testach.

Parametry Symulacji	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8
Dystans Zarażenia	3	1	2	4	5	1	3	2
Czas do Zarażenia	60	1	2	30	10	15	40	5
Zaraźliwość Patogenu	100	0	50	75	20	40	60	10
Średni Okres Inkubacji	3	1	2	1	3	1	2	3
Procent Noszący Maseczki	100	0	0	50	80	20	70	40
Skuteczność Maseczek	70	0	30	50	90	10	60	30
Liczebność Populacji	150	10	30	100	50	20	120	80
Procent Zarażonych	50	1	5	30	10	2	20	8
Odporność Populacji	90	0	50	70	30	20	80	40
Długość Symulacji	10	1	2	5	8	1	7	3
Średni czas wykrycia	3	1	2	1	3	1	2	3

5.3 Wykryte i usunięte błędy

Zidentyfikowane i skorygowane błędy obejmują:

- Po pierwszym cyklu dnia i nocy, stwierdzono kompletną utratę maseczek. Problem wynikał z braku wywołania funkcji *ShowMask()* w metodzie inicjującej symulację po zakończeniu nocy.

- Wystąpiły problemy z rezerwacją i zwalnianiem miejsc siedzących, gdzie agenci często próbowali zająć te same miejsca. Problem ten został poprzez zmianę logiki stojącej za mechanizmem. Zamiast przechowywania zajętego miejsca przez człowieka, teraz to siedzenie przechowuje informacje, o tym który człowiek na nim siedzi.
- Po przeprowadzeniu rozmowy, agenci przestawali mieć jakiekolwiek interakcje z innymi. Spowodowane to było omyłkowym wywołaniem metody zmniejszającej dystans zarażenia w funkcji obsługującej rozmowę.

Podsumowanie i wnioski

Celem pracy było opracowanie aplikacji symulującej rozprzestrzenianie się zarażeń, ze szczególnym uwzględnieniem środowiska biurowego. Projekt zakładał dostarczenie narzędzia umożliwiającego szeroką parametryzację symulacji, realistyczny model symulacyjny, uwzględnienie codziennych zachowań ludzkich w biurze oraz wizualizację rozwoju epidemii.

Praca nad projektem symulacyjnym *Infektosym* przyniosła satysfakcjonujące wyniki, spełniające niemal wszystkie postawione wymagania funkcjonalne. Pomimo osiągnięcia sukcesu w realizacji większości celów, pojawiły się pewne ograniczenia, szczególnie związane z liczbą populacji. Symulowanie dużej liczby osób w biurze, przekraczającej 100-150, okazało się niepraktyczne, co stanowi obszar do dalszych rozważań.

- **Osiągnięte cele i wymagania**

W ramach pracy udało się z powodzeniem zrealizować założone cele oraz spełnić postawione wymagania funkcjonalne. Skonstruowany model symulacyjny pozwolił na wierne odwzorowanie procesu rozprzestrzeniania się chorób w środowisku biurowym.

- **Ograniczenia i potencjalne rozwinięcia**

Pojawiły się pewne ograniczenia związane głównie z liczbą symulowanych osób. W celu lepszej skalowalności należałoby rozważyć alternatywne scenariusze lub większe mapy. Dodatkowo, rozbudowa symulacji zachowań ludzkich oraz zwiększenie liczby scenariuszy to obszary, które mogą znacząco wzbogacić funkcjonalność aplikacji.

- **Ewentualne kierunki rozwoju**

Przyszłe prace nad projektem mogłyby skupić się na ulepszaniu symulacji zachowań ludzkich, wprowadzając bardziej złożone scenariusze i interakcje między agentami. Dodatkowo, warto rozważyć rozbudowę funkcjonalności o nowe scenariusze, co umożliwi bardziej wszechstronne testowanie i analizę zachowań w różnych warunkach. Ponadto, istnieje możliwość dodania funkcji zapisywania parametrów i wyników symulacji do pliku, co ułatwiłoby dalszą analizę oraz przechowywanie danych w celu późniejszego wykorzystania.

W świetle uzyskanych wyników, projekt *Infektosym* stanowi solidną podstawę do dalszych prac rozwojowych, mających na celu jeszcze bardziej zaawansowane modele symu-

lacyjne i bogatsze doświadczenia użytkowników.

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] BioInteractive. *Modeling Disease Spread*. URL: <https://www.biointeractive.org/classroom-resources/modeling-disease-spread> (term. wiz. 30.12.2023).
- [4] 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>.
- [5] 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>.
- [6] CDC. *The Discovery and Reconstruction of the 1918 Pandemic Virus*. Centers for Disease Control and Prevention. 2019.
- [7] 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>.
- [8] Ndemis Creations. *Plague Inc.* URL: <https://play.google.com/store/apps/details?id=com.miniclip.plagueinc> (term. wiz. 30.12.2023).

- [9] 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.
- [10] 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](https://doi.org/10.1371/journal.pone.0208775). URL: <https://doi.org/10.1371/journal.pone.0208775>.
- [11] 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>.
- [12] 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).
- [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](https://doi.org/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 zakaźności 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.

λ średnia dzienna ilość zarażeń

σ wskaźnik zachorowań na dzień

γ średni dzienny współczynnik zmniejszania grupy zarażonych pacjentów

β średnia dzienny współczynnik śmiertelności

K współczynnik interwencji ludzkiej

k_1 miara izolacji fizycznej

k_2 zdolność przyjęcia do szpitala

k_3 zdolność leczenia

Źródła

```
1      private void OnTriggerEnter2D(Collider2D collider)
2      {
3          humanScript otherHuman = collider.GetComponent<
4              humanScript>();
5          int conversation = UnityEngine.Random.Range(0,100);
6          if (conversation == 1)
7          {
8              otherHuman.Conversation();
9              Conversation();
10         }
11         if(otherHuman.GetStatus() == Status.INFECTED && status
12             ==Status.HEALTHY)
13         {
14             ShowRange();
15             timeEnter = Time.time;
16         }
17         if(status == Status.INFECTED && otherHuman.GetStatus()
18             == Status.HEALTHY)
19         {
20             ShowRange();
21         }
22     }
```

Rysunek 1: Funkcja obsługująca początek kontaktu między agentami.

```
1  private void OnTriggerExit2D(Collider2D collider)
2  {
3      humanScript otherHuman = collider.GetComponent<
4          humanScript>();
5      if(otherHuman.GetStatus() == Status.INFECTED && status
6          ==Status.HEALTHY)
7      {
8          timeExit = Time.time;
9
10         float contactDuration = timeExit - timeEnter;
11
12         if (contactDuration > timeToInfection)
13         {
14             int exposed = UnityEngine.Random.Range(0,100);
15             if(exposed < virusSpreadFactor)
16             {
17                 status = Status.EXPOSED;
18                 body.color = Color.yellow;
19                 simInterface.IncreaseExposed();
20                 CalculateInfection();
21             }
22         }
23     }
24     HideRange();
25 }
```

Rysunek 2: Funkcja obsługująca koniec kontaktu między agentami.

```
1  private void CalculateInfection()
2  {
3      int infectionProbabilty = (int)(virusSpreadFactor*(1 -
4          immunity));
5      int attempt = UnityEngine.Random.Range(1,100);
6      if(attempt < infectionProbabilty)
7      {
8          infectionTime = clock.GetHoursPassed() + UnityEngine
9              .Random.Range(incubationPeriod-24,
10                  incubationPeriod+24);
11          willBeInfected = true;
12      }
13      else
14      {
15          infectionTime = clock.GetHoursPassed() + 24;
16          willBeInfected = false;
17      }
18  }
```

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()
26 {
27     if(currentActivityEndTime < clock.GetHour())
28     {
29         ReleaseSeat();
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:

- *InfektoSymulatorSource.zip* - spakowany folder zawierający pliki z kodem źródłowym
- *infektosym-installer-1.0.exe* - instalator aplikacji *InfektoSym*

Spis rysunków

1.1	Schemat działania modelu K -SEIR[9]	4
1.2	Demonstracja działania aplikacji: (a) dostosowywanie niestandardowego wirusa, (b) wprowadzenie wirusa w rzeczywistego świata, (c) wirus widoczny w rozszerzonej rzeczywistości z panelem informacyjnym [16].	9
1.3	Wygląd aplikacji „ <i>Modeling Disease Spread</i> ” [3].	10
2.1	Schemat działania modelu użytego w <i>InfektoSym</i>	16
3.1	Mapa biura: 1. Biurka do pracy, 2. Kuchnia ze stolikami, 3. Miejsce wypoczynku	20
3.2	Ekran początkowy aplikacji.	22
3.3	Ekran podczas działającej symulacji: 1. Suwaki do ustawiania parametrów, 2. Przyciski kontrolujące 3. Aktualne statystyki i czas, 4. Wizualizacja	23
3.4	Ekran zakończenia symulacji	23
4.1	Sekwencyjny diagram algorytmu kontaktu między agentami	28
4.2	Sekwencyjny diagram algorytmu symulacji zachowań ludzkich	29
1	Funkcja obsługująca początek kontaktu między agentami.	43
2	Funkcja obsługująca koniec kontaktu między agentami.	44
3	Funkcja wyliczająca szansę na rozwinięcie się choroby.	44
4	Funkcja wyliczająca szansę na rozwinięcie się choroby.	45

Spis tabel

1.1	Opis modelu epidemiologicznego K - $SEIR$ opracowany na podstawie[9]. . .	5
5.1	Parametry symulacji użyte w poszczególnych testach.	32