1. Implementacja

Poniżej zawarto opis implementacji wybranych algorytmów i obiektów symulacji.

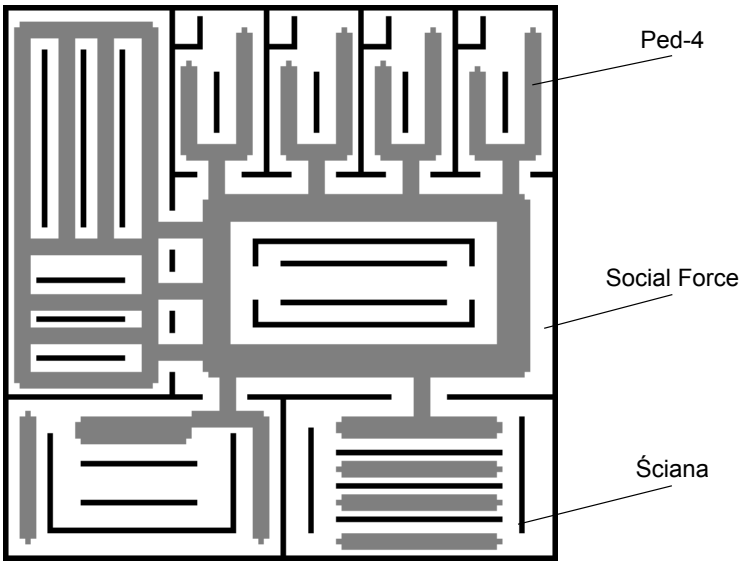
1.1. Użyte technologie i narzędzia

Symulacja została napisana w języku **Java** (wersja 7), przy czym interfejs graficzny stworzono z użyciem standardowej biblioteki Javy - **Swing**. W celu udostępnienia funkcjonalności nagrywania przebiegu symulacji do pliku *AVI* posłużono się biblioteką **Monte Media Library**, która rozpowszechniana jest na licencji *Creative Commons Attribution 3.0*.

1.2. Reprezentacja centrum handlowego

Centra handlowe reprezentowane są za pomocą map bitowych w formacie *BMP*, w których każdy piksel odpowiada jednej komórce siatki w modelu. Umożliwia to tworzenie planów obiektów handlowych w łatwy i szybki sposób, z użyciem ogólnodostępnych narzędzi jak *MS Paint* albo *GIMP*.

Każde centrum handlowe opisywane jest przy użyciu dwóch plików - mapy rozkładu pomieszczeń i algorytmów ruchu fazy operacyjnej (Rys. 1.1) oraz mapy rozkładu stref specjalnych (Rys. 1.2).



Rysunek 1.1: Przykładowy rozkład pomieszczeń i algorytmów ruchu małego centrum handlowego.

1.2.1. Mapa rozkładu pomieszczeń

Mapa rozkładu pomieszczeń określa dostępność komórek dla agentów - komórki niedostępne to ściany, stoiska rozłożone w pasażach i inne obiekty blokujące przejście. Dodatkowo zawiera ona informacje o typie algorytmu ruchu wykorzystanego w fazie operacyjnej, związanego z poszczególnymi komórkami.

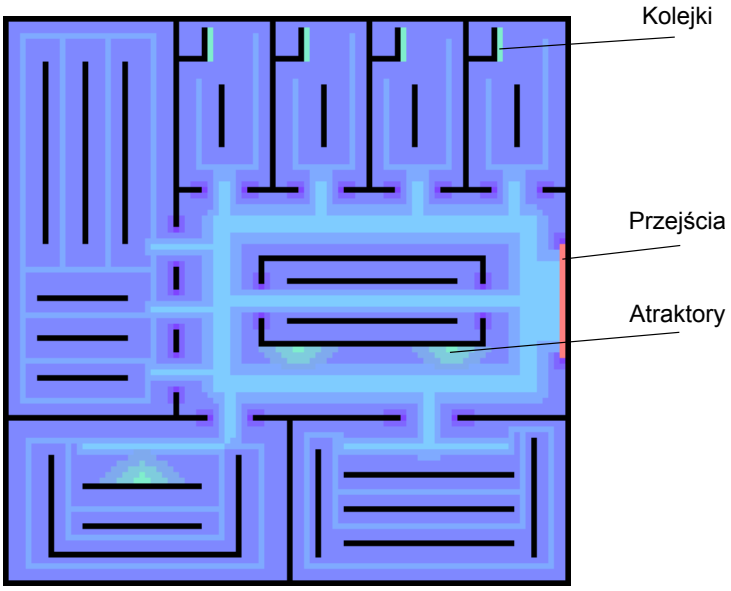
Do kodowania informacji użyto tylko jednego z trzech kanałów *RGB* - czerwonego (wartości pozostałych kanałów są pomijane). Znaczenie poszczególnych wartości kanału czerwonego dla pixeli mapy:

– $0x00$ - komórka niedostępna (obiekt uniemożliwiający przejście)

– $0x7F$ - komórka dostępna, ruch z użyciem algorytmu Ped-4.

– $0xFF$ - komórka dostępna, ruch z użyciem algorytmu Social Distances.

Na rysunku 1.2 przedstawiono przykładowy rozkład stref specjalnych małego centrum handlowego.



Rysunek 1.2: Przykładowy rozkład stref specjalnych małego centrum handlowego.

1.2.2. Mapa rozkładu stref specjalnych

Mapa rozkładu stref specjalnych centrum handlowego zawiera informacje o jego cechach funkcjonalnych - rozmieszczeniu miejsc przeznaczonych do czekania, kolejek przy kasach sklepowych, okien wystawowych. Oprócz tego mapa zawiera informacje o „sieci routingu" ułatwiającej przemieszczanie się w centrum handlowym. Wszystkie powyższe dane wykorzystywane są przede wszystkim w fazie taktycznej, gdzie są służą jako dodatkowa heurystyka algorytmu wyszukiwania ścieżek A*.

Znaczenie poszczególnych wartości kanałów *RGB* pixeli mapy ($*$ - wartość dowolna):

– $(0x7F, A, H)$ - uogólniony atraktor o *atrakcyjności* A i *czasie wstrzymania* H.

– $(0xFF, *, *)$ - wejście/wyjście

1.2.3. Uwagi

Choć w niniejszej pracy ograniczono się do przeprowadzania symulacji dla jednej kondygnacji centrum handlowego, to jednak zaprezentowane rozwiązanie jest skalowalne - symulację ruchu w galerii wielopoziomowej można potraktować jako parę równoległych symulacji (odpowiadających poszczególnym piętrom). Zjawiska obserwowane w pobliżu schodów ruchomych - grupowania się ludzi w oczekiwaniu na wejście na schody - występują również w okolicy wyjść z galerii, stąd traktując schody jak wejście-wyjście (pomiędzy piętrami) otrzymamy zbliżone rezultaty.

1.3. Ruch agentów - faza taktyczna

1.3.1. Wybór puntków docelowych

Punkty docelowe reprezentują cele podróży agentów. Algorytm wybiera zadaną ilość miejsc znajdujących się we wnętrzu centrum handlowego, które następnie sortuje w celu minimalizacji łącznej długości dróg łączących ze sobą kolejne punkty. Opisany sposób wyboru miejsc docelowych pozwala symulować wszystkie zachowania poruszających się ludzi na poziomie lokalnym (np. kolejkowanie się) i większość schematów zachowań tłumu na poziomie globalnym (z wyłączeniem pewnych bardziej złożonych zjawisk, jak różne stosunki ilościowe agentów danego typu w różnych częściach centrum handlowego).

Ponieważ w pracy postanowiono skupić się na aspekcie operacyjnym zagadnienia poruszania się pieszych po centrum handlowym, algorytm wyboru punktów docelowych został znacząco uproszczony. W praktyce punkty wybierane są w sposób całkowicie losoy.

1.3.2. Znajdowanie ścieżek

W celu wyznaczania ścieżek prowadzących do punktów docelowych wykorzystano algorytm **A***. A* jest algorytmem kompletnym, służącym do wyznaczania najkrótszej drogi łączącej dwa wierzchołki grafu. Oznacza to, że znajdzie on optymalną drogę i zakończy działanie który, jeśli tylko takowa istnieje. A* wykorzystuje heurystykę, która służy szacowaniu kosztu ścieżki prowadzącej z danego wierzchołka do celu, starając się rozpatrywać tylko najbardziej obiecujące wierzchołki.

Algorytm przochowuje zbiory wierzchołków „otwartych” (rozpatrywanych) i „zamkniętych” (tych, które już zostały sprawdzone). Działanie algorytmu rozpoczyna się od dodania aktualnej pozycji agenta do zbioru wierzchołków otwartych. Następnie algorytm wykonuje iteracyjnie następujące kroki:

– Ze zbioru wierzchołków otwartych wybierz najbardziej obiecujący wierzchołek x, minimalizujący funkcję $f(x)$:

$$f(x) = g(x) + h(x)$$

gdzie $g(x)$ określa długość drogi prowadzącej z wierzchołka początkowego do wierzchołka x, a $h(x)$ określa przewidywaną przez heurystykę długość drogi łączącej wierzchołek x i cel podróży.

– W przypadku osiągnięcia punktu docelowego zakończ działanie (sukces).

– W przeciwnym przypadku dodaj wierzchołek x do zbioru wierzchołków zamkniętych, a sąsiadujące z nim wierzchołki (wierzchołki osiągalne w jednym kroku zaczynając w x), które nie należą do zbioru wierzchołków zamkniętych, dodaj do zbioru wierzchołków otwartych.

Zależnie od zastosowanej heurystyki algorytm A* jest w stanie dostarczać optymalne ścieżki łączące dwa punkty lub suboptymalne ścieżki, które są obliczane szybciej. Ponieważ w rzeczywistym świecie pieszy praktycznie nie jest w stanie wybrać w pełni optymalnej drogi, w implementacji zastosowano złożoną, dopuszczalną (ang. *admissible*) heurystykę dostarczającą ścieżki suboptymalne. Heurystyka ta została opisana w następnej sekcji.

1.3.3. Dewiacja ścieżek

Dewiacja ścieżek zachodzi podczas działania algorytmu wyszukiwania ścieżek opisanego w poprzedniej sekcji celem ułatwienia zachodzenia pewnych zachowań, takich jak poruszanie się w odpowiedni sposób po korytarzach centrum handlowego. Odpowiednia dewiacja ścieżek została osiągnięta przez modyfikację heurystyki zastosowanego algorytmu A* - każdy fragment mapy rozkładu stref specjalnych centrum handlowego posiada przypisaną mu wartości określające parametry **uogólnionego atraktora** - *atrakcję* A oraz *czas wstrzymania* H.

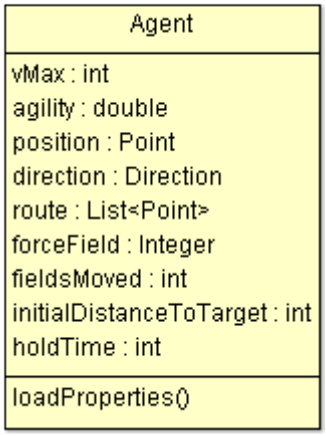
Dodatkowo heurystykę obarczono wagą $\epsilon = 5$, co pozwala szybko generować ścieżki o pożądanym koszcie (tj. takim, który nie przekraczacza ϵ razy kosztu ścieżki optymalnej).

Atrakcja odpowiada za przyciąganie agentów do pewnych stref centrum handlowego. Umożliwia ona realizację zdefiniowanych przy analizie problemu atraktorów. Jej wartość kodowana jest przez kanał zielony (G) pixeli oznaczających atraktory na mapie rozkładu stref specjalnych centrum handlowego. Wartość G należy podzielić przez $0x7F(127)$ w celu przeskalowania jej do przedziału $[0, 2)$. Heurystyka algorytmu A* wykorzystuje tę wartość, jako dodatkowy mnożnik szacowanego kosztu ścieżki: wartość 0 oznacza najsilniejszą atrakcję, wartość 1 - brak atrakcji, natomiast wartość 2 oznacza odpychanie (algorytm wyszukiwania ścieżek będzie preferował inne drogi).

Czas wstrzymania odpowiada za opóźnianie agentów podczas wykonywania ruchu. Gdy agent znajdzie się w komórce obarczonej czasem oczekiwania, jego dalszy ruch zostaje wstrzymany na zadaną ilość cykli symulacji. Wartość czasu wstrzymania kodowana jest przez kanał niebieski (B) pixeli oznaczających atraktory na mapie rozkładu stref specjalnych centrum handlowego.

1.3.4. Reprezentacja agentów

Na chwilę obecną agenci opisywani są przez parametry wpływające tylko i wyłącznie na ich ruch w fazie operacyjnej. Docelowo planowane jest rozszerzenie ich zachowania o preferencje dotyczące odwiedzanych miejsc, czasu spędzanego w galerii itp.



Rysunek 1.3: Klasa agenta

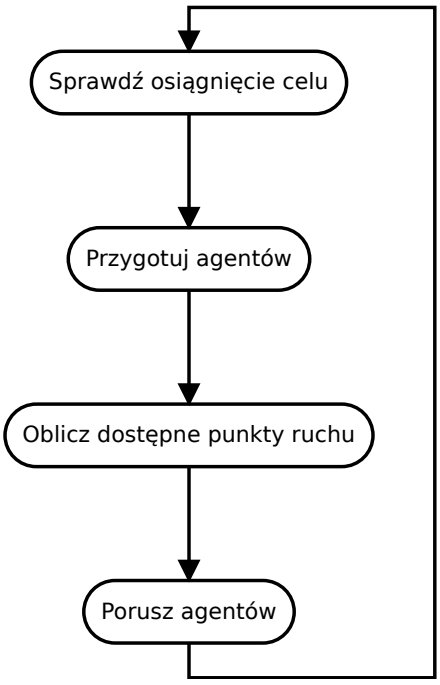
Znaczenie poszczególnych parametrów:

- **vMax** – maksymalna ilość punktów ruchu do wykorzystania w danej iteracji fazy operacyjnej
- **agility** – skłonność do dokonywania zamian z innymi podczas etapu poruszania agentów
- **position** – położenie agenta na planszy galerii
- **direction** – aktualny kierunek ruchu (dostępne kierunki są zgodne ze stronami świata)
- **route** – lista współrzędnych płytek-celów do odwiedzenia
- **forceField** – opis wartości potencjału pól wokół agenta (przy założeniu, że agent zwrócony jest w kierunku północnym)
- **fieldsMoved** – odległość (mierzona w płytkach) przebyta od ostatniego celu do danej chwili
- **initialDistanceToTarget** – teoretyczna minimalna odległość od poprzedniego do obecnego celu
- **holdTime** – pozostały czas oczekiwania (np. w kolejce)

Część spośród powyższych parametrów agenta inicjalizowana jest na podstawie plików zawierających opisy pewnych ogólnych charakterystyk ludzi, np. "typ dynamicznyóznacza człowieka poruszającego się szybko i skłonnego do dokonywania zamian, "typ emerytaóznacza stateczny krok i niechęć do zamian.

1.3.5. Ruch agentów i Ped-4

Faza operacyjna składa się z czterech wykonywanych w pętli etapów, co zilustrowano na poniższym schemacie:



Rysunek 1.4: Pętla fazy operacyjnej

Sprawdzanie osiągnięcia celu służy wyznaczeniu kolejnego celu, o ile poprzedni został osiągnięty. Cel można osiągnąć na dwa sposoby: wchodząc bezpośrednio na płytkę o zadanych współrzędnych bądź wchodząc na płytkę w sąsiedztwie celu (w tym przypadku prawdopodobieństwo uznania celu za osiągnięty zależy od odległości płytki od niego). Po wyznaczeniu nowego celu aktualizowane są parametry wykorzystywane przez algorytmy ruchu, jak

początkowa (teoretyczna) minimalna odległość od celu.

Przygotowanie agentów polega na wywołaniu dla każdego z nich metody *prepare()* algorytmu przypisanego do płytki, na której agent aktualnie się znajduje.

Obliczanie dostępnych w danym obiegu fazy iteracyjnej punktów ruchu to przepisanie aktualnej prędkości maksymalnej każdego z agentów.

W etapie poruszania agentów agenci rozpatrywani są cyklicznie. W jednej iteracji ("kroku") każdy z agentów może wykonać tylko jedną akcję, np. czkać bądź przemieścić się o jedno (!) pole. Dzięki takiemu rozwiązaniu żaden z agentów nie jest faworyzowany przy dostępie do pól.

Algorithm 1 Faza operacyjna

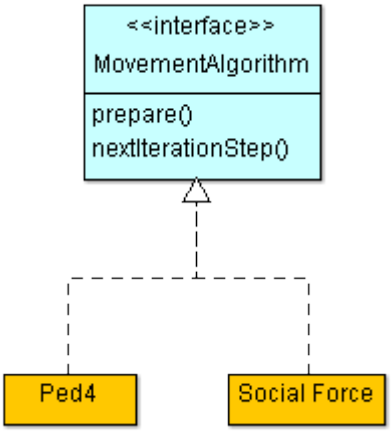
```
procedure OPERATION_PHASE
    for all agent w galerii do                                  ▷ sprawdź osiągnięcie celu
        if agent osiągnął cel OR agent uznał, że osiągnął cel then
            pobierz kolejny cel z listy
            zapisz statystyki dotyczące dalszej drogi
        end if
    end for

    for all agent w galerii do                                  ▷ przygotuj agentów
        call Algorithm.prepare with agent
    end for

    for all agent w galerii do                                  ▷ oblicz punkty ruchu
        punkty_ruchu_agenta ← predkosc_agenta
    end for

    for krok od 1 do v_max do                                   ▷ porusz agentów
        for all agent w galerii do
            if agent jest wstrzymywany then
                zmniejsz pozostały czas wstrzymania
            else if agent nie wykorzystał jeszcze w tym kroku punktów ruchu then
                call Algorithm.nextIterationStep with agent
                call Feature.performAction with agent
                wykorzystaj punkt ruchu
            end if
        end for
    end for
end procedure
```

Metoda *performAction()*, wywoływana dla elementów galerii, ustawia parametry agenta związane z symulacją takich zdarzeń, jak np. oczekiwanie w kolejce.

Wszystkie algorytmy ruchu implementują interfejs *MovementAlgorithm*. Interfejs ten zawiera dwie metody wywoływane w fazie operacyjnej dla każdego agenta:

– *prepare()* - wywoływana jeden raz, na początku; służy ustawieniu agenta na odpowiedniej pozycji początkowej

– *nextIterationStep()* - wywoływana tyle razy, ile agent posiada punktów ruchu; obsługuje ruch "do przodu"; przy każdym jej wywołaniu agent przesuwa się co najwyżej o jedno pole (w sąsiedztwie Moore'a)



Rysunek 1.5: Interfejs *MovementAlgorithm* i jego realizacje

1.3.6. Social Force

Poniżej przedstawiono pseudokody metody odpowiedzialnej za ruch "do przodu"w algorytmie Social Force:

Funkcja wyznaczająca dostępne pola ogranicza możliwoś ruchu tak, aby agent albo dalej przemieszczał się po dotychczasowej trajektorii, albo skręcił w stronę celu.

Potencjał pola obliczany jest na podstawie siły oddziaływania innych ludzi (ujemna składowa) oraz odległości pola od celu (dodatnia składowa). W przypadku, gdy agent już dość długo zmierza do celu (tzn. przebyta przez niego droga znacząco przekracza minimalną drogę konieczną do osiągnięcia celu), lecz nie może go osiągnąć, przestaje zwracać taką uwagę na innych agentów – efekt ten został osiągnięty przez dodatkowe zwiększenie potencjału dostępnej płytki leżącej najbliżej celu.

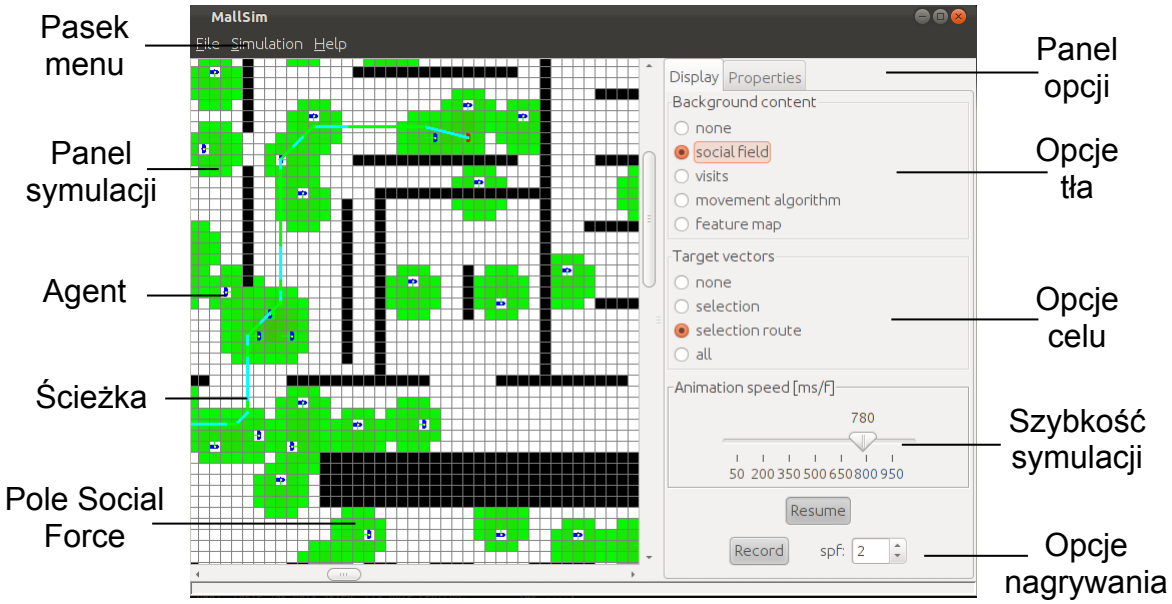
W przypadku, gdy brak jest dostępnych płytek, agent obraca się odrobinę w miejscu, by w następnym kroku zyskać nowe możliwości ruchu (ze względu na zmianę zbioru dostępnych płytek).

Algorithm 2 Ruch agenta w algorytmie Social Force

```
procedure NEXTITERATIONSTEP
    if brak możliwości ruchu then
        obróć agenta                                   ▷ umożliwia przeszukanie nowych obszarów
    else
        cel ← wyznacz_pole_o_najwiekszym_potencjale()
        przesuń agenta na docelową płytkę
        odzwierciedl kierunek wykonanego ruchu poprzez zmianę zwrotu agenta
    end if
end procedure

function WYZNACZ_POLE_O_NAJWIEKSZYM_POTENCJALE
    dostepne_pola ← wyznacz_dostepne_pola()
    for all dostępne pole do
        uwzględnij wartość pola potencjału innych ludzi i odległość od celu
    end for
    if agent jest zmęczony then
        zwiększ dodatkowo potencjał dostępnego pola najbliżej celu
    end if
    pozostaw pola o najwyższym potencjale
return pole leżące najbliżej celu
end function

function WYZNACZ_DOSTEPNE_POLA
    pola_agenta ← pola w sąsiedztwie von Neumanna leżące w półkolu wyznaczonym przez kierunek ruchu agenta
    pola_celu ← pola w sąsiedztwie von Neumanna leżące w półkolu wyznaczonym przez kierunek do celu
return pola_agenta AND pola_celu                       ▷ część wspólna zbiorów
end function
```

1.3.7. Wizualizacja i pozyskiwanie danych

Wizualizację przebiegu symulacji wykonano w oparciu o bibliotekę **Java Swing** z racji na prostotę implementacji i zadowalające efekty wizualne, jakie ona dostarcza. Na rysunku 1.6 zaprezentowano wygląd głównego okna programu podczas przykładowej symulacji.



Rysunek 1.6: Główne okno programu.

Program udostępnia wiele konfigurowalnych opcji odpowiadających za działanie symulacji i jej wizualizacji dostępnych za pośrednictwem *panelu opcji*:

- **Opcje tła** - wybór danych wizualizowanych w tle symulacji; do wyboru jedna wartość z: *brak*, *gradient Social Force*, *rozkład odwiedzin danych komórek centrum handlowego*, *rozkład algorytmów ruchu*, *rozkład stref specjalnych centrum handlowego*.
- **Opcje celu** - opcje wizualizacji celu podróży agentów; do wyboru jedna wartość z: *brak*, *wektor najbliższego celu wybranego agenta*, *aktualna ścieżka wybranego agenta*, *wektory najbliższego celu wszystkich agentów*.
- **Szybkość symulacji** - pasek wyboru szybkości symulacji (ilość milisekund na krok symulacji); wartości z przedziału 50 do 1000 milisekund.
- **Opcje nagrywania** - opcje odpowiadające za nagrywanie przepiegu symulacji; obecnie dostępna jest tylko konfiguracja ilości kroków na klatkę filmu (*spf*).

...oraz *paska menu*:

- **Seed** - opcje odpowiadające za *ziarno* symulacji umożliwiające kontrolowanie powtarzalności wyników.

W celu ułatwienia pozyskiwania danych do późniejszej analizy program udostępnia zakładkę *Properties* generującą dynamicznie podgląd parametrów wybranego agenta oraz możliwość nagrania przebiegu symulacji za pomocą biblioteki **Monte Media Library** - przebieg symulacji zapisywany jest w głównym katalogu programu w formacie *.avi*.

Bibliografia

[BT86] A. Borgers and H. Timmermans. A model of pedestrian route choice and demand for retail facilities within inner-city shopping areas. *Geographical Analysis*, 1986.

[FSE99] Iso/tr 13387-1:1999 fire safety engineering. 1999.

[Hel92] D. Helbing. A fluid dynamic model for the movement of pedestrians. *Complex Systems*, 1992.

[HM94] D. Helbing and P. Molánr. Social force model for pedestrian dynamics. 1994.

[KB04] K. Kitazawa and M. Batty. Pedestrian behaviour modelling - an application to retail movements using a genetic algorithm. 2004.

[WGM06] J. Wąs, B. Gudowski, and P. Matuszyk. Social distances model of pedestrian dynamics. *Unknown Journal*, 2006.

[Woo02] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons Ltd, 2002.