Dokumentacja projektu  
Agents in the Harbour  
Lilianna Hełminiak, Maciej Kaim, Wojciech Kuprianowicz

Spis treści

[Krótki opis programu 1](#_Toc359837052)

[Przeznacznie programu 2](#_Toc359837053)

[Przegląd zastosowań programu 2](#_Toc359837054)

[Moduły 2](#_Toc359837055)

[Main 2](#_Toc359837056)

[Map 2](#_Toc359837057)

[Field 2](#_Toc359837058)

[Crate 3](#_Toc359837059)

[Crane 3](#_Toc359837060)

[Forklift 3](#_Toc359837061)

[Ship 3](#_Toc359837062)

[Display 3](#_Toc359837063)

[Message 3](#_Toc359837064)

[Agenci 3](#_Toc359837065)

[Dźwig 3](#_Toc359837066)

[Wózek widłowy 3](#_Toc359837067)

[Statek 3](#_Toc359837068)

[Protokoły komunikacji 3](#_Toc359837069)

[Wiadomości 3](#_Toc359837070)

[Negocjacje 3](#_Toc359837071)

## Krótki opis programu

Agents in the Harbour to program symulujący proces załadowania statku w porcie przez dźwigi, które mają ograniczony zasięg ramienia oraz przez mobilne wózki widłowe. Jest to program wykorzystujący systemy inteligentnych agentów, gdzie każdy wózek i dźwig to agent mający możliwość komunikowania się z innymi agentami.

Celem agentów jest załadowanie statku w jak najszybszy sposób wykonując jak najmniej ruchów.

## Przeznacznie programu

## Przegląd zastosowań programu

## Moduły

### Main

Główny program uruchamiający działanie programu.

#### Opis funkcjonalny modułu

Moduł jest przeznaczony do uruchomienia programu z właściwymi parametrami.

Zawiera jedynie konstruktor w którym tworzy obiekt map oraz wyświetla go.

### Map

Moduł zawierający wszytkie konfiguracje programu.

#### Opis funkcjonalny modułu

Moduł jest przeznaczony do przechowywania wszystkich informacji na temat wyświetlanej mapy.

Map zawiera tablicę dwuwymiarową która symuluje port w 2D. Elementami tablicy są obiekty typu field.

Moduł zawiera następujące procedury:

**stopThreads(): void** – zatrzymuje wszystkie wątki aby program mógł wstrzymać działanie

**inMapBounds(pos): bool** – sprawdza, czy jakieś pole jest w obszarze mapy

**distance(coordinates1, coordinates2): int** – wylicza odległość między dwoma współrzędnymi.

**commonArea(crane1, crane2): ((int y, int x), height, width)** – wylicza wspólne pole o kształcie prostokąta dla dźwigów crane1 i crane2. Zwraca trójkę: współrzędne lewej górnej kratki, wysokość oraz szerokość wspólnego pola.

**commonStorageFields(crane1, crane2): [(int y, int x)]** – zwraca listę współrzędnych wspólnych pól dźwigów crane1 i crane 2 które są typu Storage, posortowaną po odległości od crane1.

**drawMap() : void** – wyświetla mapę

Map korzysta z innych modułów: crane, ship, forklift, display oraz field. Jest to potrzebne, ponieważ w tym module są tworzeni agenci.

#### Sytuacje niepoprawne: W-Ku TODO

Kontrola poprawności danych

### Field

Moduł który określa typ danego pola na mapie.

#### Opis funkcjonalny modułu

Moduł jest przeznaczony do przechowywania informacji o danym polu na mapie, czyli jakiego jest typu oraz jakie obiekty się na nim aktualnie znajdują.

Może być następujących typów:

STORAGE\_TYPE – na polu takiego typu można układać paczki,

ROAD\_TYPE – po takiego typu polach mogą się przemieszczać wózki widłowe,

CRANE\_TYPE – na polach tego typu znajdują się dźwigi,

SHIP\_TYPE – na tych polach znajdują się pola należące do statku

Field zawiera listę, na której są przechowywane obiekty które leżą na danym polu.

Zawiera następujące procedury:

**isStackable(): bool** – sprawdza, czy można na danym polu położyć paczkę

**countCrates(): int** – zwraca liczbę obiektów mieszczących się na danym polu

**getAllCratesIds(): [int]** – zwraca id wszystkich paczek mieszczących się na polu

**isCratePresent(crateId): bool** – sprawdza, czy dana paczka znajduje się na liście obiektów pola

**getCratePosition(crateId): int** – jeśli paczka jest obecna na liści obiektów danego pola to jest zwracana jej pozycja

**getTopCrateId(): int** – zwraca id paczki leżącej na szczycie

**removeCrateFromTop(): crate** – ściąga ostatnią paczkę na liście obiektów pola i ją zwraca.

**putCrateOnTop(crate): void** – dołącza do listy obiektów na danym polu kolejną paczkę.

**getCrane(): Crane** – zwraca obiekt Crane który jest na danym polu.

**isForkliftPresent(): bool** – sprawdza, czy zawiera obiekt Forklift.

Moduł korzysta z modułu Crate w celu korzystania z informacji o id paczki jaka znajduje się na polu.

#### Sytuacje niepoprawne

W poszczególnych procedurach modułu, przed wykonaniem akcji jest sprawdzane jakiego typu jest pole. Jeśli typ nie zgadza się z daną akcją, to wywoływany jest wyjątek. Np. nie można zdobyć informacji o paczce, która miałaby leżeć na polu CRANE\_TYPE.

### Crate

Moduł który zawiera specyfikacje paczki mieszczącej się na mapie.

### Crane

Klasa która specyfikuje obiekt dźwigu.

### Forklift

Klasa która specyfikuje obiekt wózka widłowego.

### Ship

Klasa która specyfikuje obiekt statku.

### Display

Moduł który zawiera szczegóły dotyczące grafiki programu.

### Message

Moduł który określa rodzaj przesyłanych wiadomości między agentami.

## Agenci

### Dźwig

### Wózek widłowy

### Statek

## Protokoły komunikacji

### Wiadomości

#### Statek – Agenci: Potrzebuję paczek:

#### Dźwig – dźwig: Mam dostęp do statku

### Negocjacje

#### Dźwig – dźwig: podaj paczkę

#### Dźwig – dźwig: podnieś paczkę

#### Wózek – wózek: gdzie jesteś?

#### Wózek – dźwig: podaję paczkę