Symulacja wieloagentowego systemu robotycznego

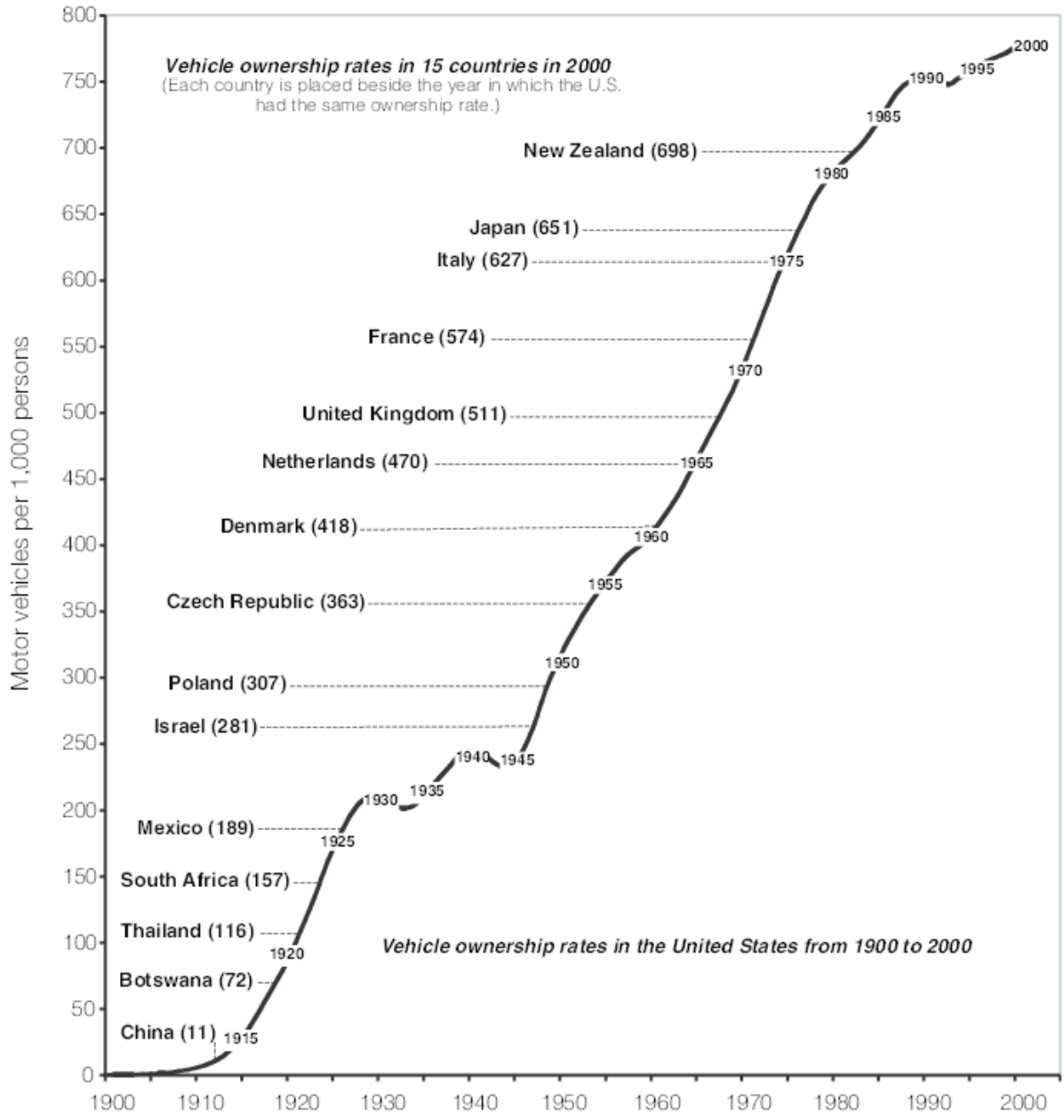
Opis symulacji

Zakres zastosowań robotów we współczesnym świecie staje coraz szerszy: roboty zamieniają ludzi na fabrykach, dostawią jedzenia na dom, patrolują ulicy.

Nie mniej jednak, współczesne roboty mają taki problem, że one wykonują tylko jedno dość proste zadanie w jednym momencie, co poważnie ogranicza ich możliwości. Jednym z rozwiązań tego problemu jest tworzenie systemów składających się z kilku robotów, w których każdy robot wykonują jedną ściśle określoną rolę. To podejście nabyło dość szerokiej popularności w kołach akademickich i jest dość aktywnie w nich rozwijane [1].

Niestety w rzeczywistości ono nie jest dość szeroko stosowane ze względu na wysoką cenę robotów, co tak naprawdę nie jest poważnym ograniczeniem, ponieważ progres ciągle idzie do przodu i rzeczy które wcześniej były za

drogie stają się dostępne (na przykład samochody, które najpierw były tylko dla osób bogatych, a teraz prawie każda druga osoba ma swój samochód)



Wykres liczby właścicieli samochodów w Stanach w okres od 1900 do 2000

Biorąc pod uwagę ten fakt, postanowiliśmy napisać symulację wieloagentowego systemu robotycznego, żeby pokazać możliwe miejsce zastosowań. Jako główny temat obraliśmy budownictwo, ponieważ uważamy, że ta dziedzina — to miejsce w którym ludzka praca nie jest konieczna.

Przed sobą postawiliśmy następujące cele:

1. Pokazać jak liczba robotów wpływa na czas budownictwa
2. Pokazać jak wybrane materiały wpływają na czas budownictwa
3. Pokazać jak rozmieszczenie budynków wpływają na czas budownictwa
4. Pokazać jak dobrane modele roboty wpływają na czas budownictwa
5. Pokazać jak landszaft wpływa na czas budownictwa

Architektura

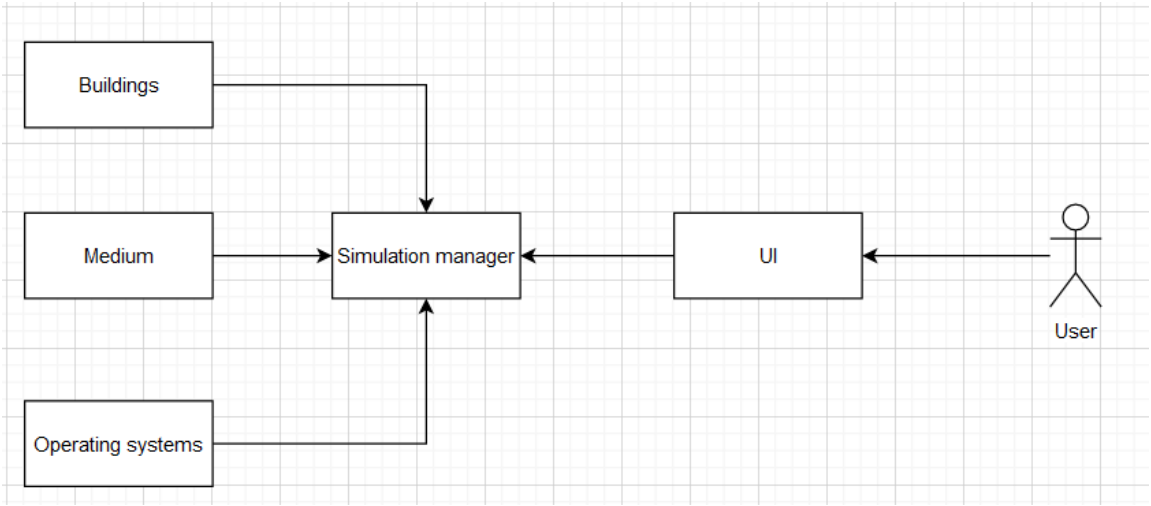
Całą symulację można podzieli na dwa podstawowe obiekty.

1. Manager symulacji

   To jest obiekt, która łączy w całą logikę symulacji oraz UI. On odpowiada za to, żeby rozpocząć symulację, stworzyć wszystkie potrzebne obiekty i zrobić podsumowanie całej symulacji.

   Obiekty na diagramie niżej:

   1. Medium — środowisko poprzez które komunikują się roboty. Odpowiada za przesyłanie danych
   2. Operating systems — system operacyjny, który jest instalowany na robota i który nim zarządza. (w rozdziale niżej będzie podane więcej informacji na ten temat)
   3. Buildings — wszystkie budynki, który użytkownik rozmieszczał na mapie i które muszą być zbudowane

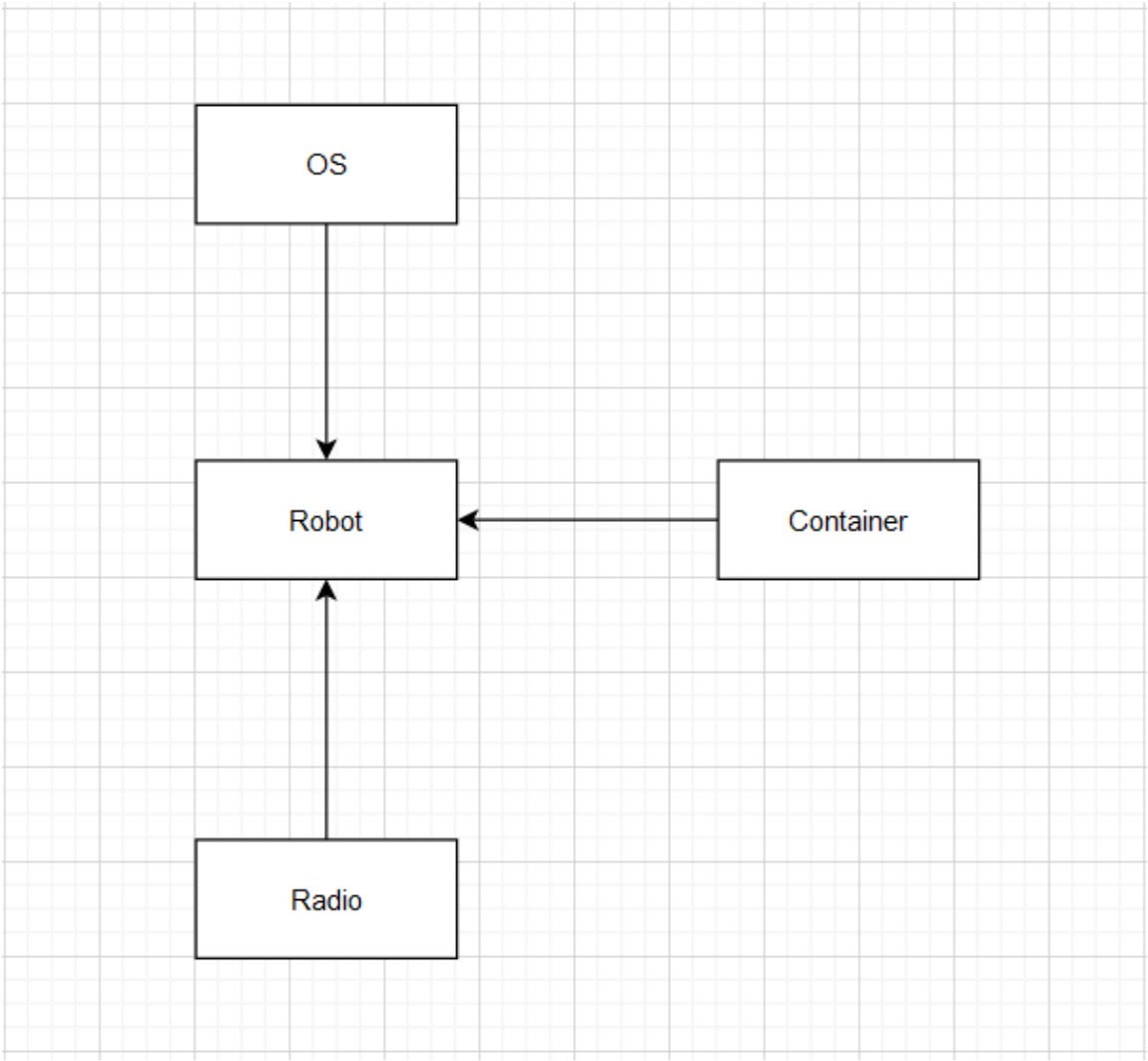


Uproszczony model managera symulacji

2. Robot

   Najgłówniejsza rzecz w całej symulacji. Robot nie prostą strukturą i jest składa się z trzech podstawowych komponentów

   1. OS — system operacyjny. O nim poniżej
   2. Container — kontener w którym robot przenosi materiały. Dla każdego robota kontenery się różnią rozmiarem, który jest ustawiany na podstawie wymiarów robotów odczytanych z ich technicznych opisów [2],[3]
   3. Radio — element który odpowiada za komunikację. (dokładniej w rozdziale niżej)



Uproszczony model robota

Fizyczne modele robotów

W świecie rzeczywistym każdy robót jest przede wszystkim ciałem fizycznym. On jest zrobiony z pewnych materiałów i ma nadany pewny kształt od tego zależy to z jaką prędkością będzie mógł się przemieszczać oraz ile materiałów będzie mógł. Nie mniej jednak to fizyczne ciało nie odpowiada w całości za to jak robot będzie się zachowywać, odpowiada za to system operacyjny który jest na niego zainstalowany. Jako przykład, podaję obrazek niżej:

Manipulatory zrobione dlatego, żeby przestawiać obiekty z taśmy na taśmę grają na fortepianie https://www.youtube.com/watch?v=tNu6PT7KKQg

Systemy operacyjne

Jak zobaczyliśmy w poprzednim rozdziale to, jak będzie się zachowywać robot całkowicie zależy od tego, jaki system operacyjny będzie na niego zainstalowany. Z tego powodu, w naszej symulacji, cała logika zachowania się robota mieści się w systemie operacyjnym.

W symulacji istnieje trzy typy systemów operacyjnych:

1. Operator

   Zadaniem robota który ma ten system operacyjny jest zarządzanie budownictwem: przypisuje budowniczych do budynków i wysyła transporterów z materiałami których potrzebuje ten lub inny budowniczy

2. Transporter

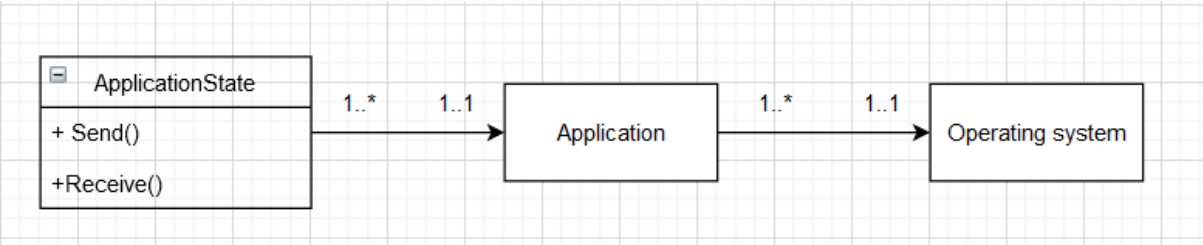
   Zadaniem robota który ma ten system operacyjny jest przenoszenie wszystkich materiałów z magazynu do budowniczych które ich potrzebują

3. Budowniczy

   Zadaniem tego robota jest budowanie budynku. On przekłada materiały z swego kontenera do budynku i wysyła wiadomość z żądaniem do operatora w przypadku jeżeli mu brakuje materiałów

Nie mniej jednak, podobnie jak w rzeczywistości, system operacyjny nie jest czymś co w całości opisuje logikę, a tylko czymś co daje dostęp do resursów

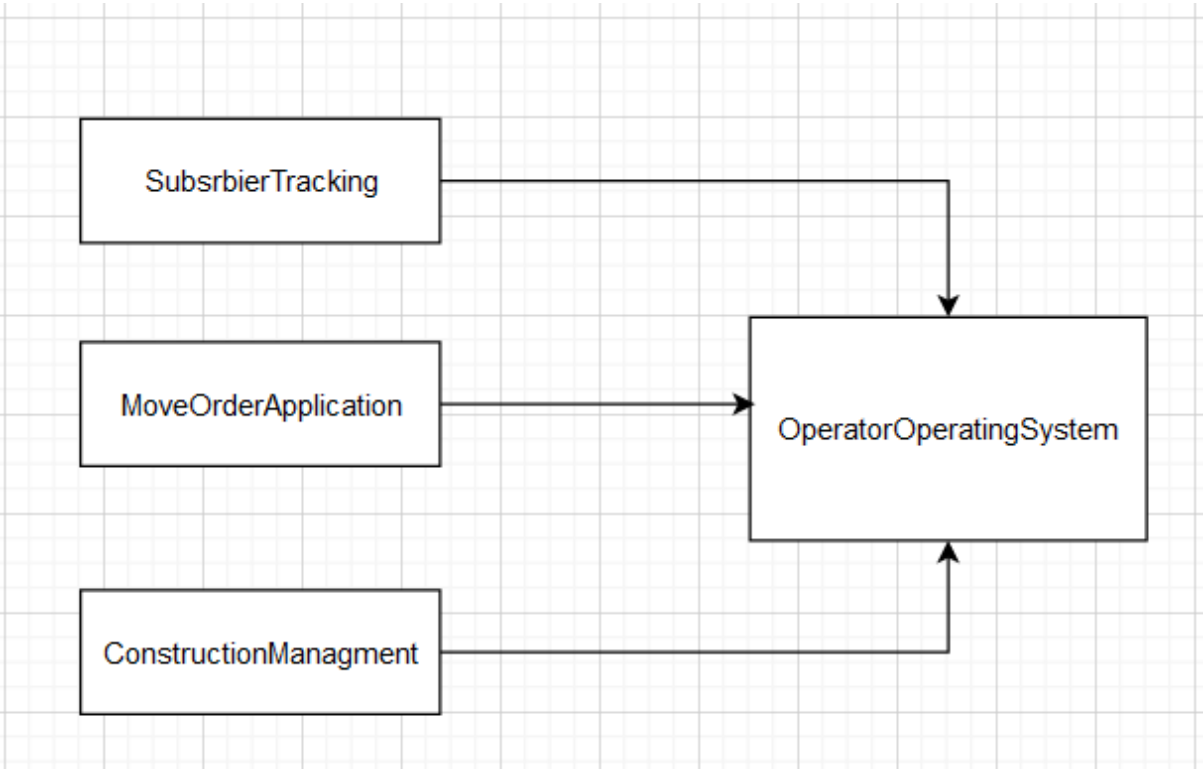
oraz pozwala zainstalować pewne aplikacje. Z tego powodu, każdy system operacyjny składa się aplikacji, jak jest pokazane na diagramie niżej. Każda aplikacja ma jakieś konkretne zadanie za które ona odpowiada, a także ma pewne stany w których może się znajdować. Od tego, w jakim stanie aplikacja się znajduje zależy to, jak ona będzie reagować na przychodzące ramki i jakie ramki będzie mogła wysłać.



W jednym systemie operacyjnym może znajdować się dużo aplikacji. Jedna aplikacja w jednym momencie przebywa tylko w jednym systemie operacyjnym. Jedna aplikacja może mieć dużo różnych stanów. W jednym momencie jedna aplikacja znajduje się tylko w jednym stanie

Niżej podaję dokładniejszy opis każdego z systemów operacyjnych oraz aplikacji które są na nich zainstalowane

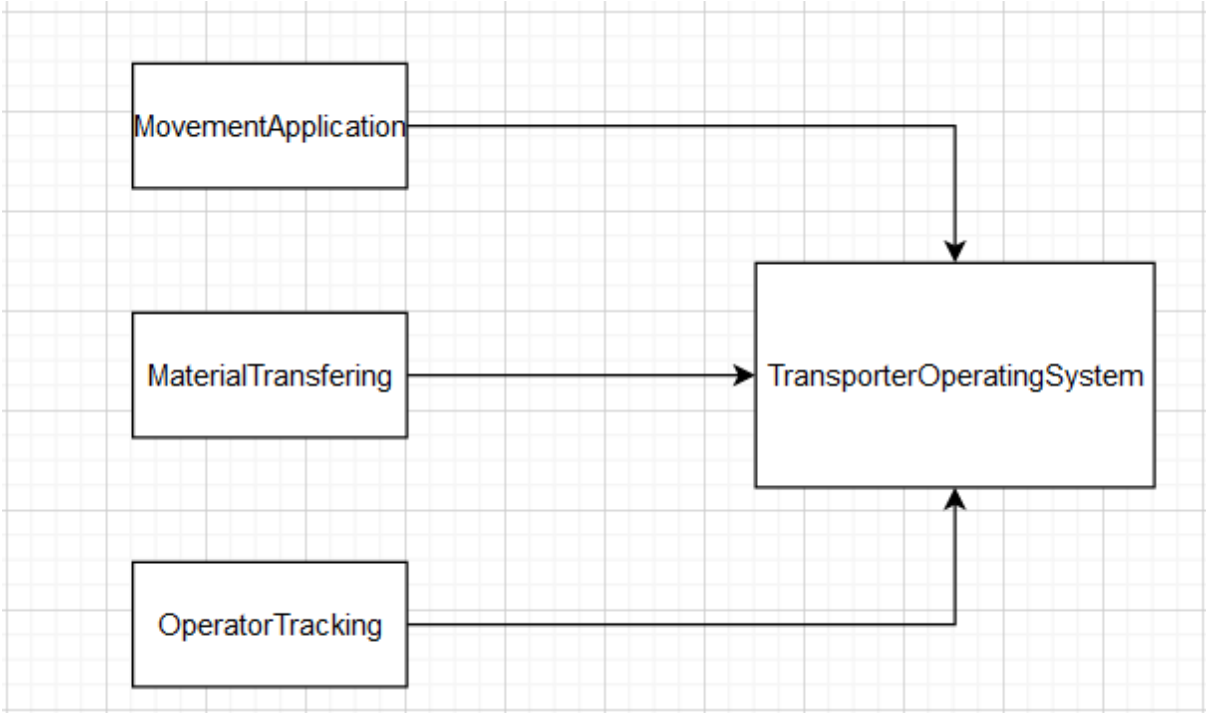
System operacyjny operatora



Aplikacje

- SubscriberTracking — aplikacja która dodaje nowych robotów do tych którymi zarządza operator
- MoveOrderApplication — aplikacja która odpowiada za to, żeby robot którym zarządza ten operator doszedł do celu
- ConstructionManagment — aplikacja która odpowiada za to, żeby budownictwo się odbywało: wysyła transporterów z niezbędnymi materiałami, przypisuje budowniczych do budynków, patroluje miejscowość

System operacyjny transportera

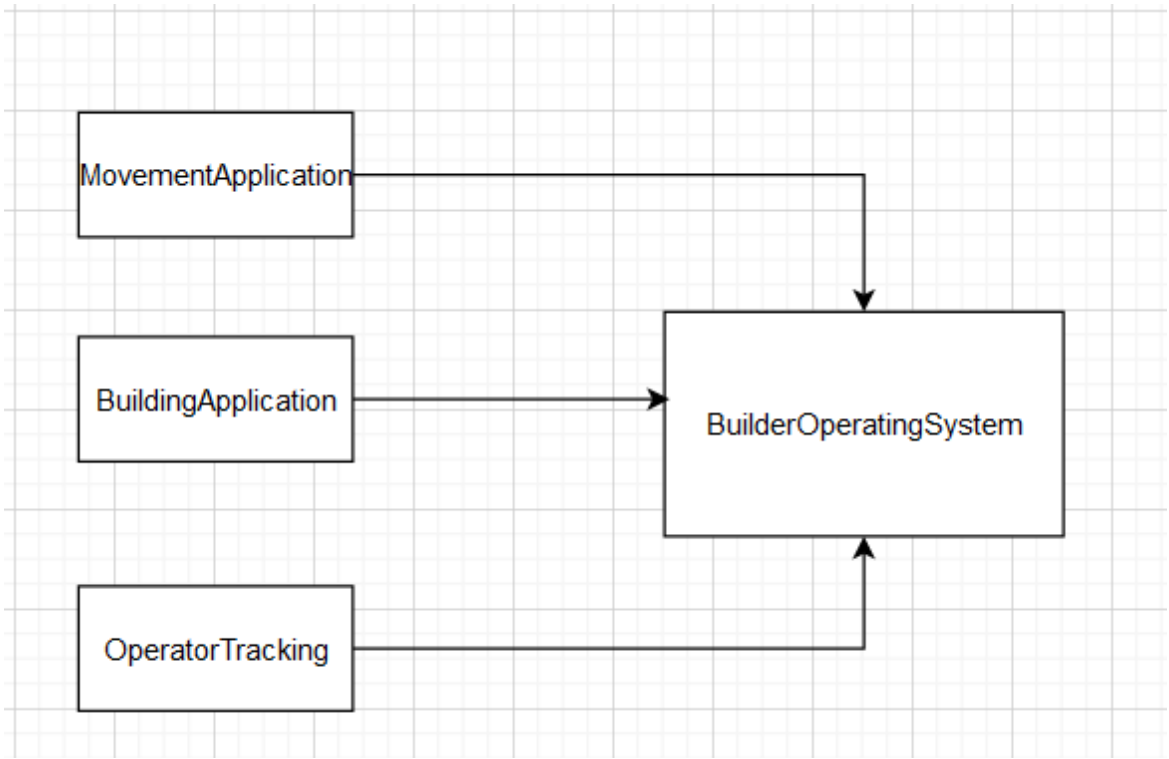


Aplikacje:

- MovementApplication — zarządza poruszaniem się robota. Zadaniem tej aplikacji jest odbieranie rozkazów od operatora, i raportowanie mu kiedy robot dojdzie do celu
- MaterialTransfering — aplikacja która odpowiada za to, jakie materiały przenosi robot. Ona kontroluje aktualny transportera: czy on może zacząć transportować nowe materiały czy nie.

- OperatorTracking — aplikacja która odpowiada za to, żeby wysyłać heartbeat do operatora potwierdzając tym samym, że ten robot nadal jest w zasięgu operatora i nadal może odbierać ramki

System operacyjny budowniczego

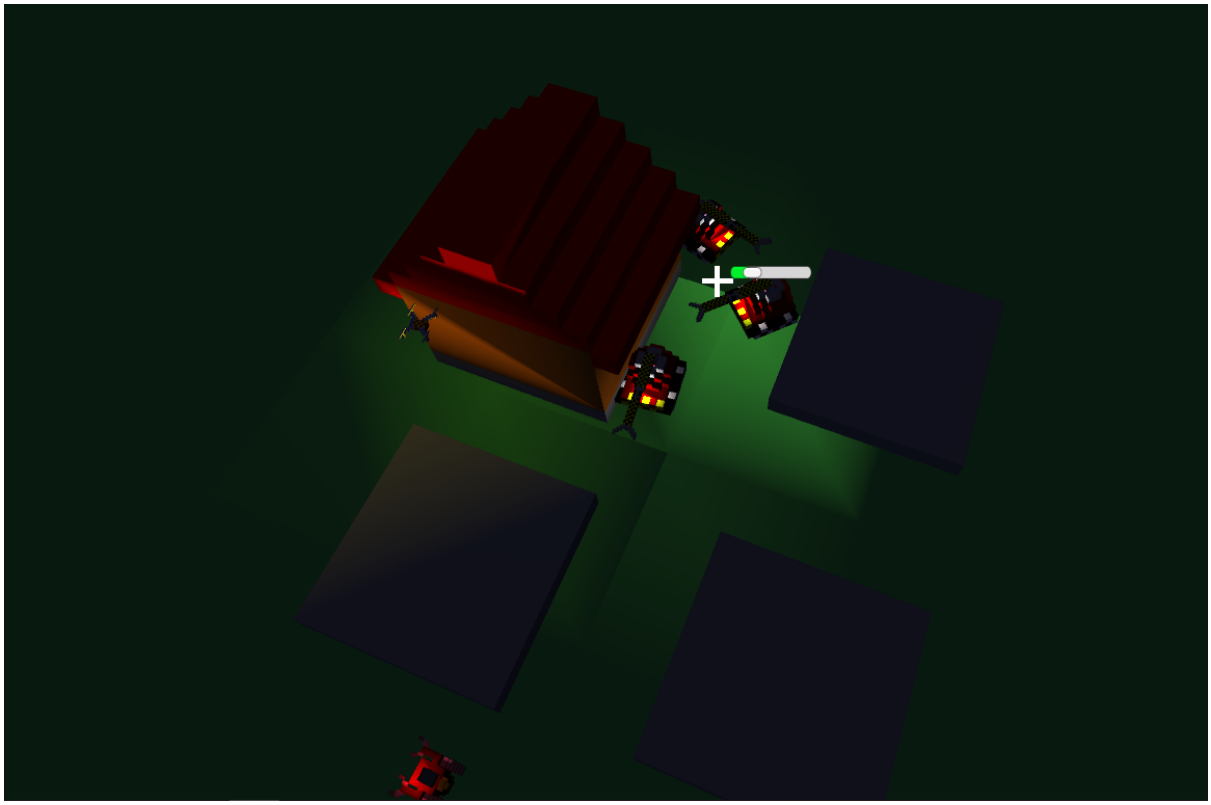


Aplikacje MovementApplication oraz OperatorTracking robią dokładnie to samo co u Trasnportera.

Aplikacja BuildingApplication odpowiada za kontrolę materiałów potrzebnych do budownictwa oraz wysyłania do operatora ramek z odpowiednim żądaniem.

Nawigacja w świecie

Wszystkie roboty poruszają się w świecie orientując się na to co widzą przed sobą i starając się uniknąć przeszkód. Ciekawym zjawiskiem związanym z nawigacją są kolizje które pojawiają się w przypadku gdy kilka robotów chce jednocześnie przejść przez jedno miejsce. Zjawisko to ma bezpośredni wpływ na to, ile czasu zajmie budownictwo.



Sytuacja w której może pojawić się kolizja

Komunikacja

W rzeczywistości istnieje kilka różnych typów komunikacji w robotycznych systemach wieloagentowych [4]:

1. Komunikacja niejawna. Jest realizowana na podstawie kamer i sensorów. Ona polega na tym, że roboty zostawiają pewne znaki w miejscach które one odwiedziły (na przykład w przypadku symulacji to może być zbudowany budynek) i potem inne roboty na podstawie tych znaków decydują co muszą robić dalej. W przypadku symulacji to wyglądało by tak:

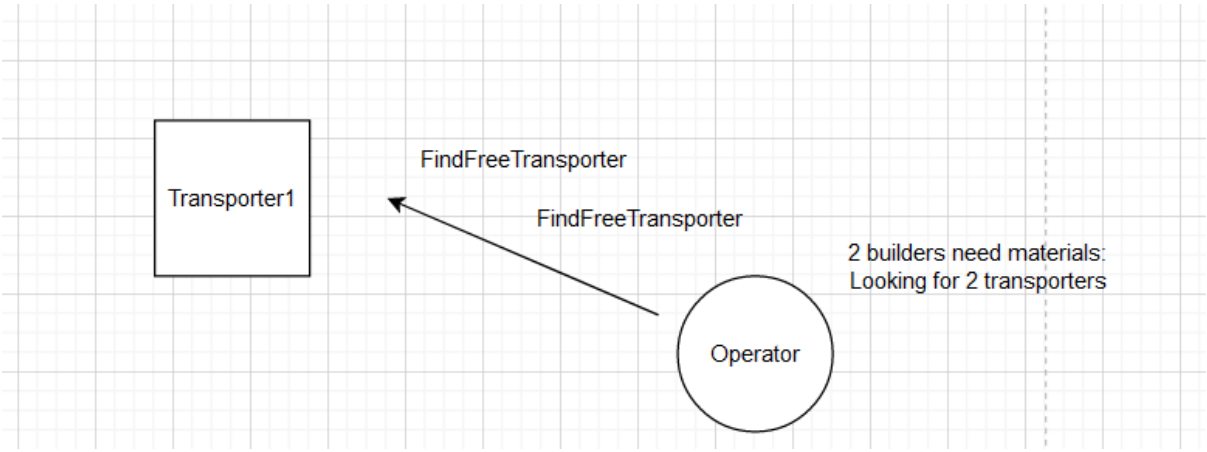
   Każdy budowniczy patroluje wszystkie budynki i w przypadku, jeżeli widzi, że ten budynek jest niezbudowany — zostaje przy nim. Podobnie robią transportery, które jeżdżą od budynku do budynku i jeżeli przy tym budynku jest budowniczy — sprawdzają, czy on potrzebuje materiałów.

   Główną wadą takiego podejścia jest to, że roboty w przypadku budownictwa na dużej przestrzeni, czas potrzebny transporterom na patrolowanie byłby dość duży, co przedłużyłoby czas potrzebny na to żeby skończyć proces. Jednym z rozwiązań takiego problemu byłoby zwiększenie liczby transporterów w taki sposób, że przy każdym budynku

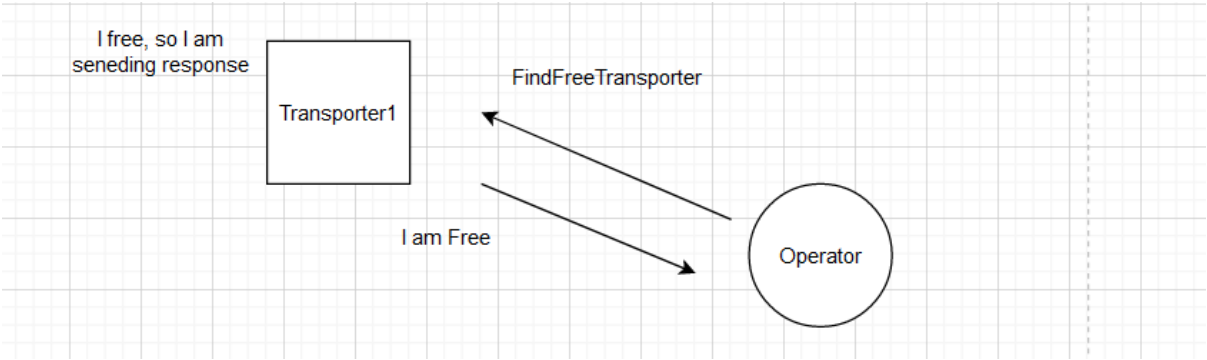
byłby jeden transporter, ale takie podejście ma taką wadę, że środki potrzebne na budownictwo gwałtownie rośną. Z tego powodu, wybraliśmy inny rodzaj komunikacji — komunikację jawną

2. Komunikacja jawna polega na tym, że roboty jawnie mówią o tym, że potrzebują czegoś. Najczęściej w takich celach jest wykorzystywana komunikacja bezprzewodowa [5]. W naszej symulacji my nie zagłębialiśmy się w takie szczegóły związane z komunikacją bezprzewodową jak czułość, path loss i różne modele propagacji fal radiowych, a przyjęliśmy założenie, że każde radio ma ustawioną promień, który opisuje z jakiej odległości mogą być przyjęte sygnały. Jedną rzeczą o której warto wspomnieć przy komunikacji bezprzewodowej jest problem związany z tym, że mogą pojawić się wyścigi. Poniższy przykład ilustruje problem:

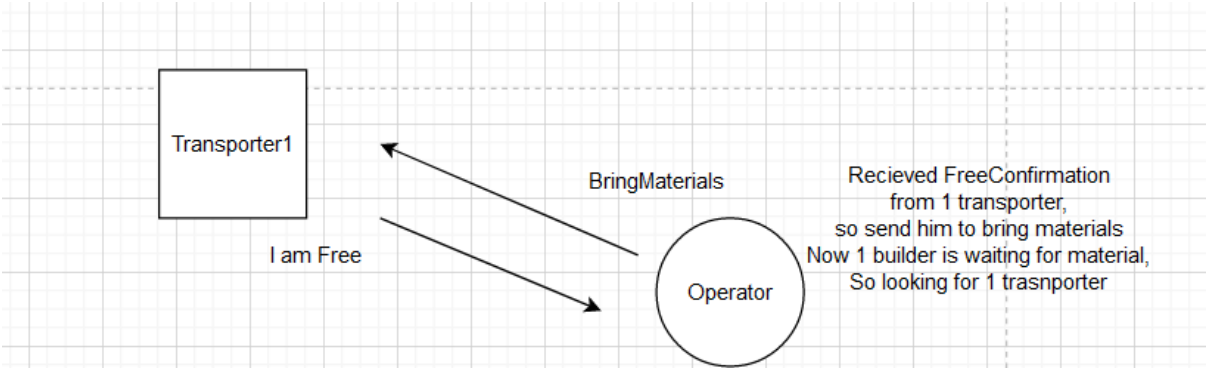
Mamy dwóch budowniczych które zgłaszają do swojego operatora, że potrzebują materiałów i przechodzą w stan oczekiwania. Dalej idą ilustracje



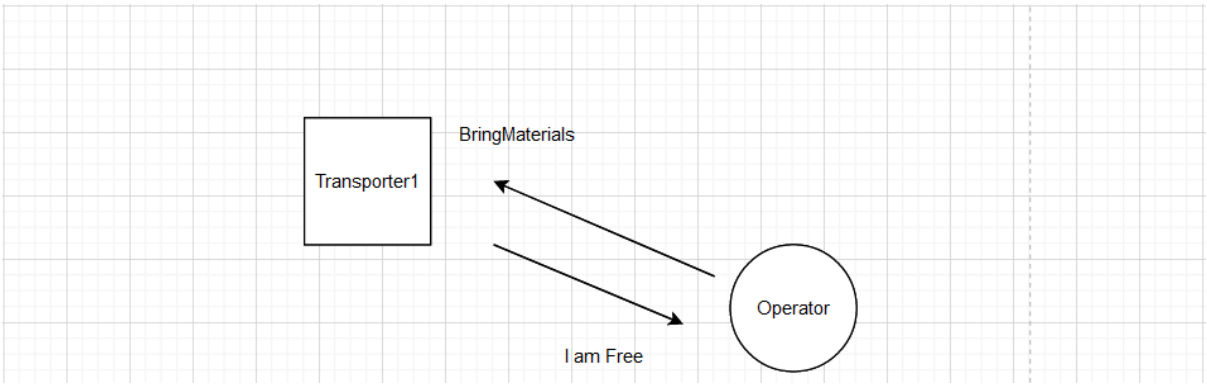
Operator odebrał dwie ramki od dwóch budowniczych z żądaniem przynieść materiały, dlatego wysyła dwie ramki które szukają wolnego transportera



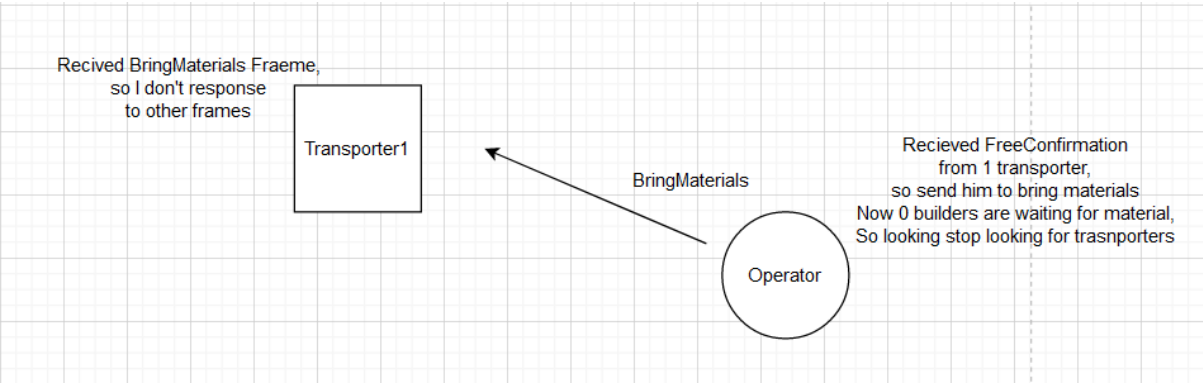
Transporter odbiera ramkę i wysyła potwierdzenie do operatora. Tym czasem jedna z dwóch wysłanych ramek nadal idzie



Operator odbiera potwierdzenie i wysyła koordynaty miejsca w które potrzeba przynieść materiały. W tym momencie uznaje również, że nie jeden z budowniczych ma już przypisanego transportera, więc nie potrzeba szukać dalej. Tym czasem transporter odebrał ramkę poprzednią ramkę , która szukała wolnych transporterów i wysyła potwierdzenie tego, że jest wolny



Ramki idą



Transporter odebrał ramkę z rozkazem od operatora i przeszedł w stan w którym on nie odbiera żadnych nowych ramek. Tym czasem operator odebrał poprzednie potwierdzenie I Am Free od Transportera, wysłał mu rozkaz z położeniem drugiego budowniczego i uznał, że znalazł wszystkich transporterów dla wszystkich budowniczych

W tym momencie, powstaje problem, ponieważ Transporter nie odbierze tą ostatnią ramkę (ponieważ w takim stanie w tej chwili się znajduje).

Oznacza to, że jeden z budowniczych ciągłe będzie czekać na materiały, ale nigdy ich nie otrzyma ponieważ operator uznał, że znalazł odpowiedniego transportera.

W związku z tym musieliśmy opracować protokół, który pozwoli tego uniknąć. Widzieliśmy trzy możliwe rozwiązania:

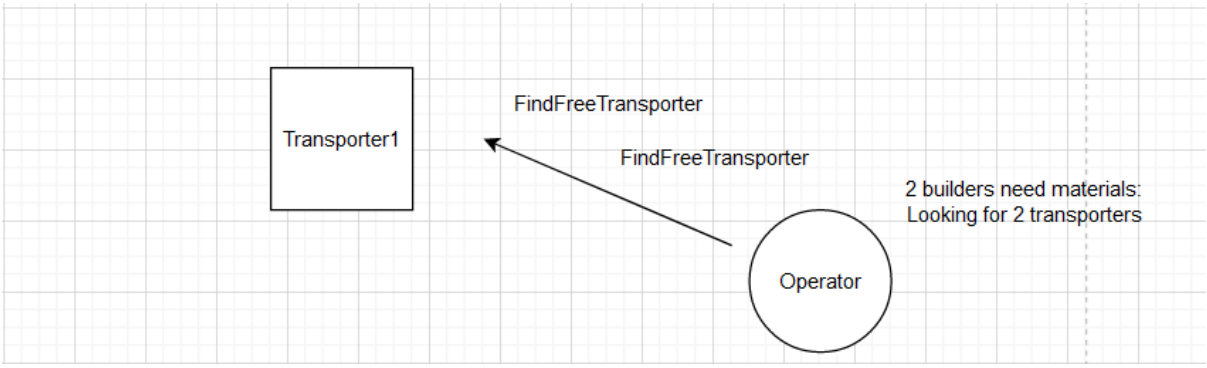
1. Budowniczy co jakiś czas wysyła żądanie do operatora póki nie otrzyma materiałów
2. Operator nie usuwa budowniczego ze swojego rejestru, aż do momentu, póki nie otrzyma od niego ramkę z potwierdzeniem odbioru materiałów.
3. Transporter ustawia się w stan `LISTENING` kiedy odbiera ramkę `Find Free Trasnporter` od operatora i zostaje w tym stanie przez pewien czas. Będąc w stanie nie odpowiada na żadne ramki oprócz rozkazu operatora.

Pierwsze podejście jest źle, ponieważ może pojawić się sytuacja w której dwa transportery zaczną transportować materiały do budowniczego (operator będzie ciągłe szukać transporterów i może zdarzyć się tak, że w momencie kiedy jeden z nich przenosi materiały operator znajdzie kolejnego transportera i wyśle go do tego samego budowniczego)

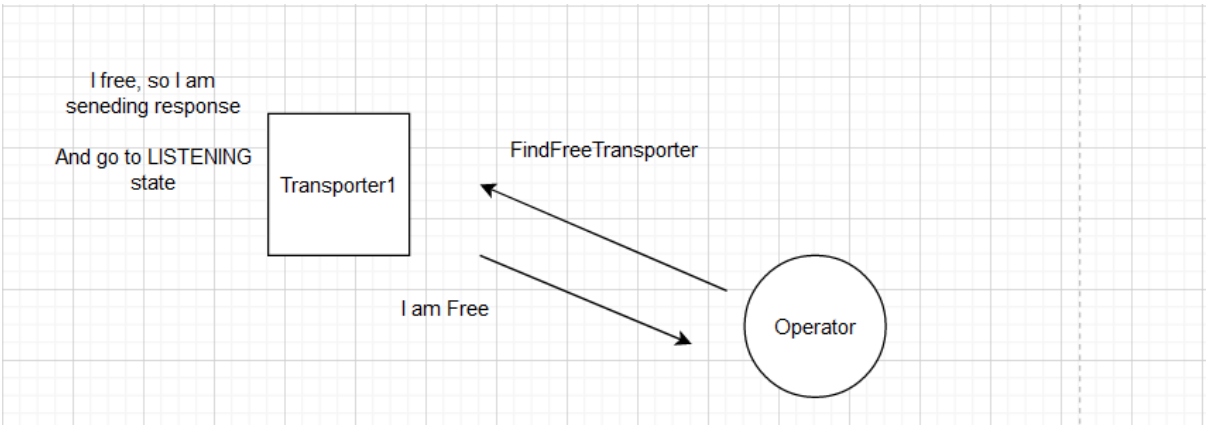
Drugie podejście jest źle z tego samego powodu co pierwsze: operator może wysłać kilka transporterów do tego samego miejsca.

Natomiast trzecie podejście jest pozbawione takich wad. Ponadto, jest ono podobne do szeroko stosowanego protokołu TCP.

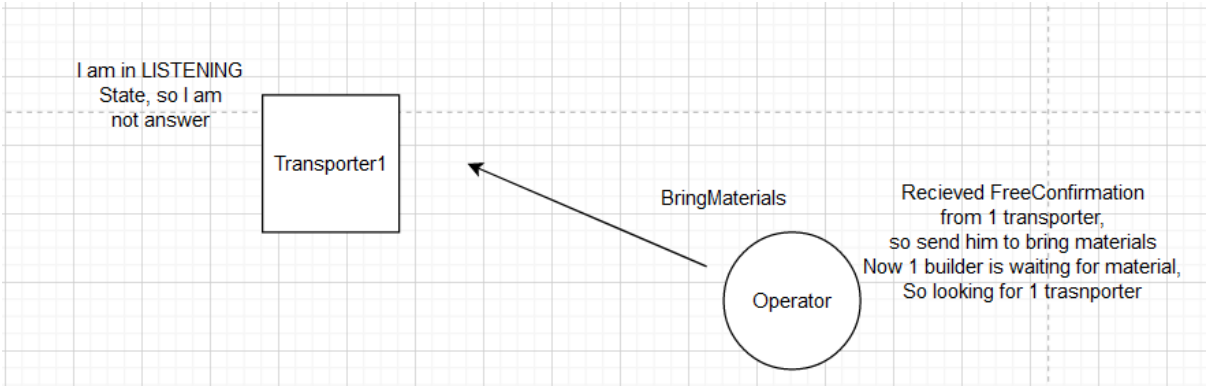
Z zastosowaniem tego protokołu powyższa sytuacja wygląda tak:



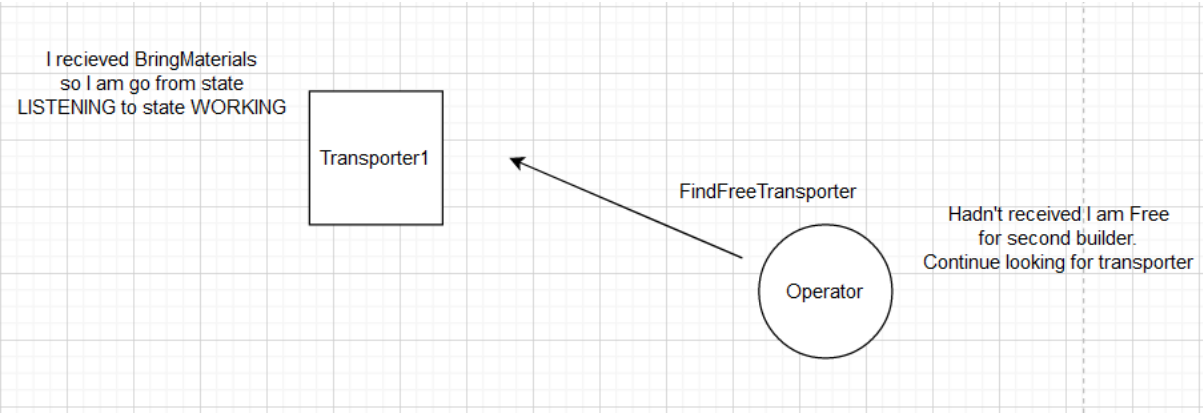
Jak poprzednio, operator wysyła dwie ramki FindFreeTransporter



Transporter odbiera ramkę FindFreeTrasnporter, wysyła potwierdzenie tego że jest wolny i przechodzi w stan Listening — nie odpowiada na żadne ramki oprócz BringMaterials



Nie odpowiada na ramkę FindFreeTransporter, ponieważ znajduje się w stanie LISTENING



Transporter odbiera ramkę BringMaterials i zaczyna pracować. Operator nadal szuka wolnego transportera dla drugiego budowniczego

Jak widzimy, ten problem zniknął i teraz każdy budowniczy otrzyma materiały na które reaguje.

Kolejną ważną rzeczą jest przyjęta struktura ramki, ponieważ od niej zależy to, która aplikacja odbierze wiadomość i jak ona na nią zareaguje.

Struktura ramki jest następująca

```
 srcMac          |           destMac
---------------------------------------
transmissionType |  destinationRole
---------------------------------------
messageType
---------------------------------------
message
---------------------------------------
payload
```

Pola:

- `srcMac` — adres źródła
- `destMac` — adres docelowy
- `transmissionType: Broadcast | Unicast` — pole które wskazuje na to, w jaki sposób potrzeba rozesłać tą ramkę. Typ jest potrzebny z tego powodu, że ramka jest tworzona w jakiejś z aplikacji, a później wysyłana do Radio, które już samo decyduje co ono musi robić dalej. (w kolejnym punkcie będzie przykład użycia)
- `destinationRole: Opeartor | Transporter | Builder` — pole, które zwykle jest stosowane w połączeniu z `transmissionType`:`Broadcast.` Wskazuje na to, dla której jakiej roli ta ramke jest boardcastowana. Przykład użycia: kiedy

roboty szukają operatora, one wysyłają ramkę z `transmissionType: Broadcast` oraz z `destinationRole:Operator`

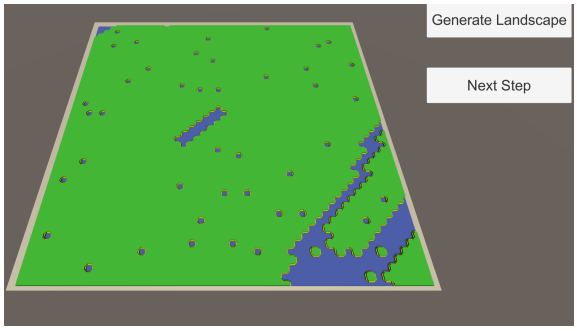
- `messageType: Service | Request | Ack | Nack` — pole, które wskazuje jak obsłużyć tą wiadomość
- `message: BringMaterials | FindFreeTransporter ...` — pole które wskazuje na to jaka to jest wiadomość
- `payload: float []` — dodatkowe dane, które moga być przekazane z ramką (na przykład położenie budowniczego dla ramki z `message:BringMaterials`

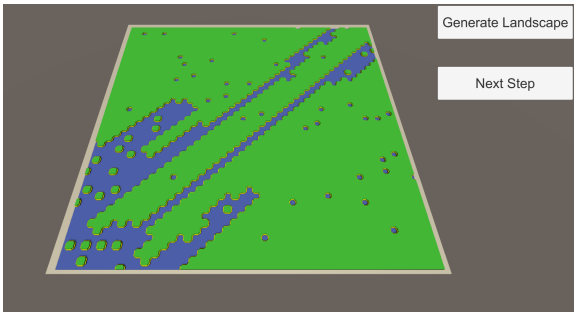
Możliwości użytkownika

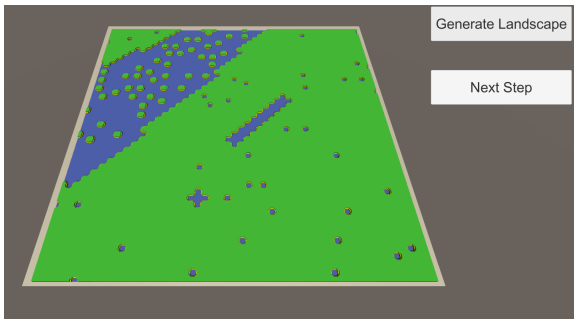
Naszym celem było dać użytkowniku możliwośc zmieniać dość dużo rzeczy, które mogą mieć wpływ na przebieg symulacji: zaczynając od wyboru tego, jakich i ile robotów przypisać do każdej z ról, kończąc tym, jak musi wyglądać teren na którym będzie się odbywać budownictwo.

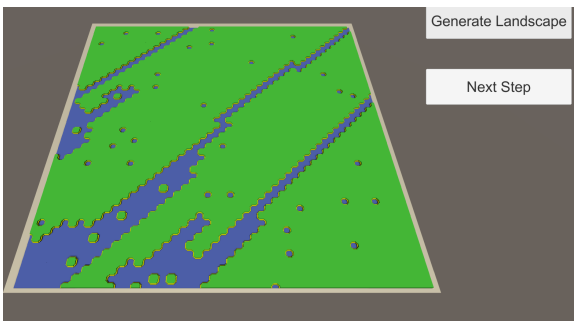
Lista konfigurowalnych rzeczy i tego jaki one mają wpływ:

- Teren— jeżeli na wygenerowanym landszafcie będzie więcej rzek, roboty będą musieli je obchodzić, co oznacza, że budownictwo będzie trwać dłużej. Przykładowe landszafty:

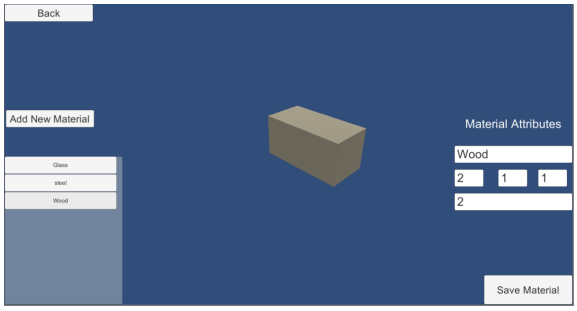




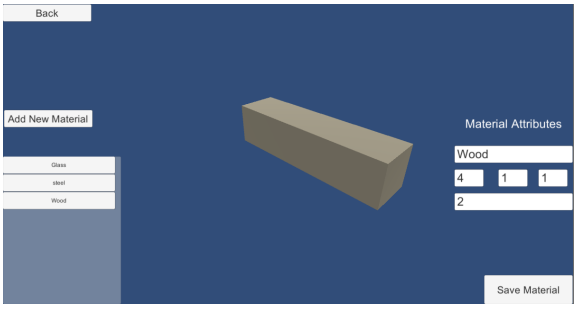




- rozmiar materiałów — każdy robot, ma kontener o ściśle określonej wielkości, co oznacza, że w zależności od tego jaki będzie rozmiar materaiłu, tyle będzie w stanie zmieszczyć w swoim kontenerze, co będzie miało wpływ na czas budownictwa

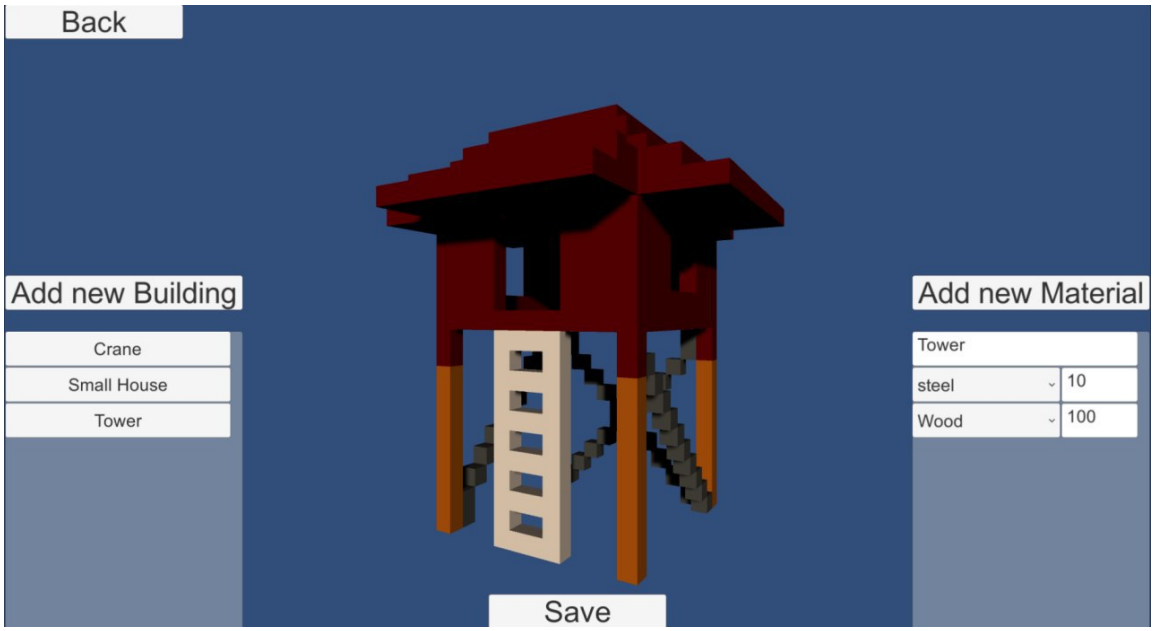


Materiał przed edycji



Materiał po edycji

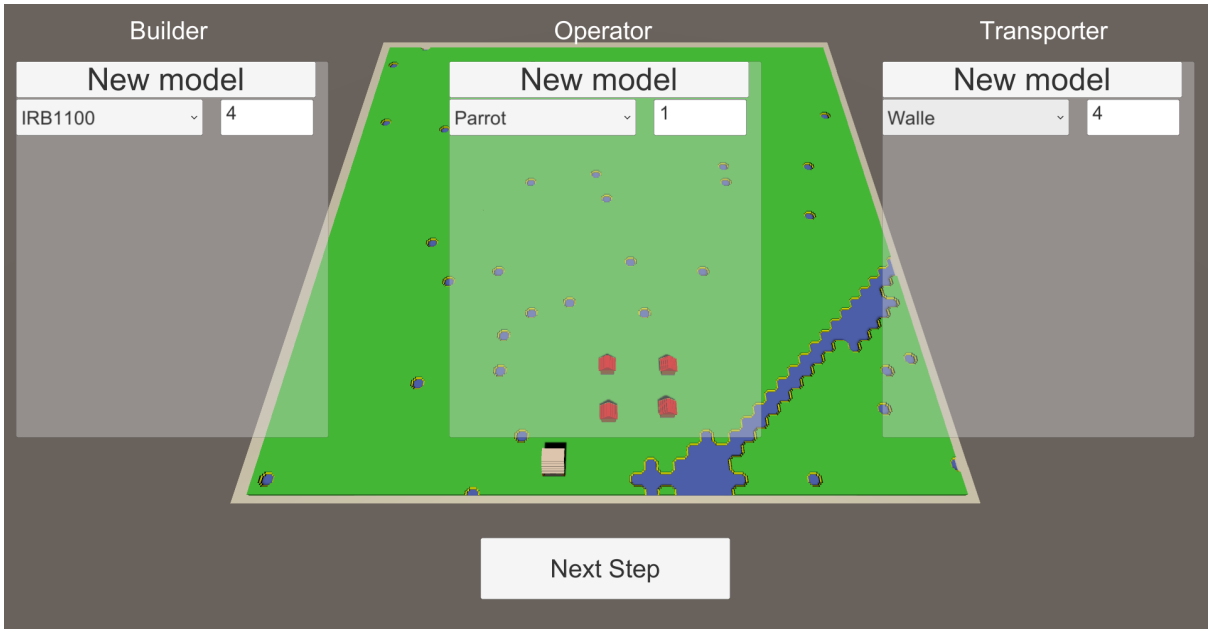
- materiały potrzebne na budowę budynku — im więcej jest wymaganych materiałów, tym więcej czasu zajmuje budownictwo



Okno edytowania materiałów

- Robotów przypisanych do poszczególnych ról: każdy robot ma swoje parametry techniczne, takie jak ilość materiałów które może przenosić,

prędkość z którą się porusza. Oznacza to, że jeżeli naprzykład przypisać do roli transportera, robota, który jest przyznaczony na rolę budowniczego (czyli takiego, który ma większy kontener, ale jest wolniejszy), to on będzie w stanie przenosić więcej rzeczy, ale wolniej, więc może pokazać się dobrze jeżeli budynki są blisko magazynu i źle, jeżeli są daleko



Okno przypisywania robotów

Różne scenariusze

Porównywanie IRB i Walle w roli transportera

IRB — to robot, który ma koneter o dużej pojemności, ale wolno się przemieszcza.

Walle — robot, który przymieszcza się szybciej, ale ma mniejszy kontener

4 IRB w role transporterów, 4 IRB w role budowniczych

4 Walle w role transporterów, 4 IRB w role budowniczych

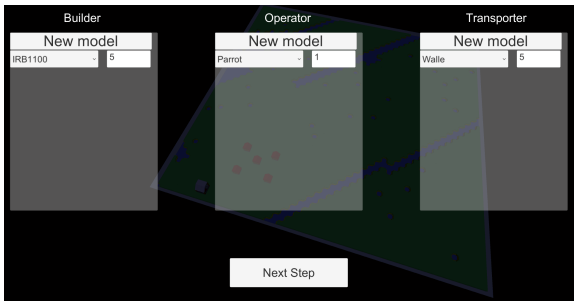
Wnioski

Kiedy Walle występuje w role transportera, budownictwo odbywa się szybciej. Czyli większy kontener nie zawsze daje przewagę

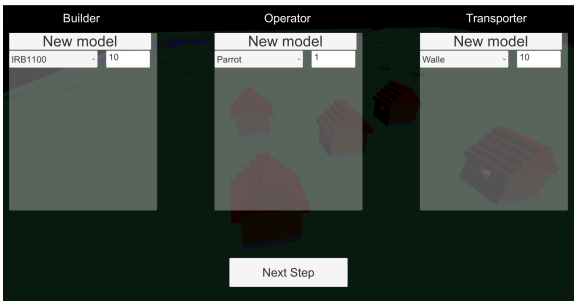
Porównanie różnej ilości transporterów

W tym przykładzie porównamy dwa przypadki:

1. Liczba transporterów i budowniczych jest taka sama jak liczba budynków
2. Liczba transporterów i budowniczych jest dwa razy większa od ilości budynków (na jeden bydunek przypada dwa budowniczych i dwa transportery)



Początkowe ustawienia



Początkowe ustawienia

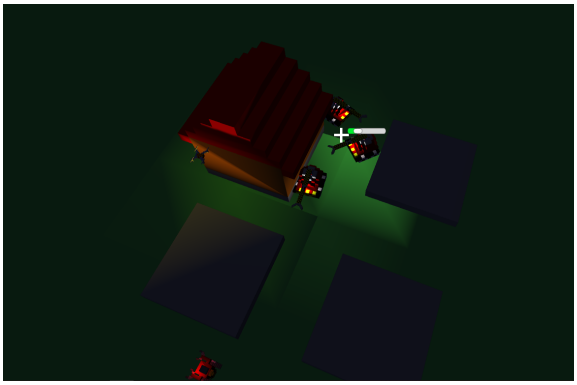
Wynik

Wynik

Wnioski

Jak widzimy, wbrew pozorom, większa liczba robotów w cale nie oznacza to, że budownictwo będzie trwać szybciej (w 1. przypadku — 0.6 dni , w 2. — 0.8 dni). Jest to związane z tym, że duża ilosć robotów na małej przestrzeni (a dokładnie taką przestrzeń my mamy: magazyn jest blisko budynków, przejścia między budynkami są małe) powoduje kolizji między robotami, co skutkuje dłuższym czasem budownictwa

Budynki były blisko, więc dużo transporterów mogły jednocześnie podjeżdzać do magazynu

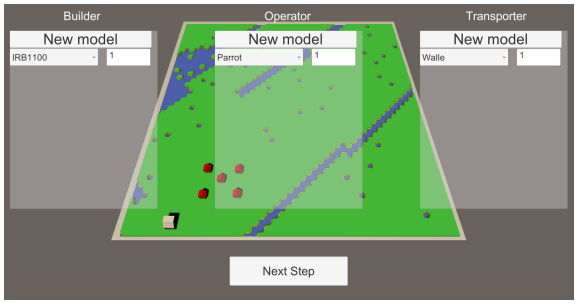


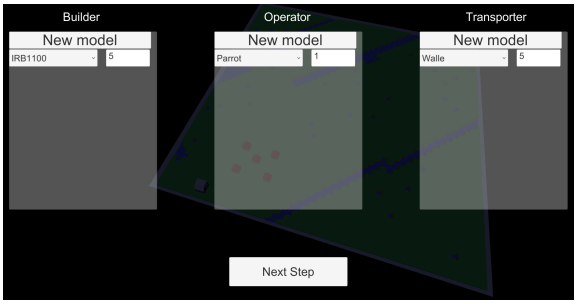
Kolejna kolizja w przejściach między budynkami

1 transporter, 1 budowniczy

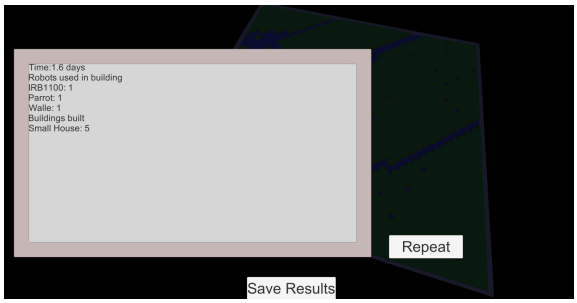
W poprzednim scenariuszu widzielismy, że mniejsza liczba robotów powoduje szybszą budowe, powodu braku kolizji. W tym scenariuszu porównamy nieco inne ustawienia:

- 1 transporter, 1 budowniczy, 5 budynków
- 5 transporterów, 5 budowniczych, 5 budynków





Początkowe ustawienia



Wynik

Początkowe ustawienia

Wynik

Wnioski

Jak widziymy, mniejsza liczba robotów w cale nie oznacza, to że budownictwo będzie trwało szybciej, musimy dobrać optymalne ustawienia

Podsumowanie

Jak widzimy, symulacja pozwala użytkowniku oszacować ustawinia, które będą najlepiej pasować do tego, co on chce zbudować, i dośc często pokazuje wnioski, które na pierwszy rzut oka nie są oczywiste: na przykład większ liczba robotów ≠ mniejszy czas budownictwa.

Symulację można rozwijać dalej, dodając do niej kolejne elementy które będą miec wpływ na pracę robotów: na przykład więcej różnych landszftów, warunki pogodwe które będą mieć wpływ na to stan robotów, genetyczne algorytmy do wyboru najlepszych ustawień dla budynków ustawionych przez użytkownika.

Źródła

[1] Zool Hilmi Ismail, Nohaidda Sariff

A Survey and Analysis of Cooperative Multi-Agent Robot Systems: Challenges and Directions

[2] https://github.com/fobannit-studio/Robolation/blob/master/Documentation/technical-specifications/robots/transporters/autonomous-vechicles-transporters.pdf

[3] https://github.com/fobannit-studio/Robolation/blob/master/Documentation/technical-

specifications/robots/operators/anary-lighwieghted-with-good-camera.png

[4] Ronald C. Arkin, Tucker Balch

Communication in Reactive Multiagent Robotic Systems

[5] Kwang-Cheng Chen, Hsuan-Man Hung

Wireless Robotic Communication for CollaborativeMulti-Agent Systems