Raport Końcowy

Inteligentne Techniki Obliczeniowe

Bienias Krzysztof

Nr indeksu: 261235

Spis treści

**Treść projektu1**

**Koncepcja projektu1**

**Wprowadzenie teoretyczne**2

Wpisz tytuł rozdziału (poziom 3)3

**Wpisz tytuł rozdziału (poziom 1)4**

Wpisz tytuł rozdziału (poziom 2)5

Wpisz tytuł rozdziału (poziom 3)6

## Treść projektu

**Projekt L3 Rachunek sytuacyjny i wnioskowanie w przód.**

Wykonać program realizujący symulację środowiska i misję ratunkową autonomicznego latającego drona (agenta) w nieznanym terenie, o cyfrowej postaci „N × N × H” „komórek”. Celem działania agenta jest odnalezienie pilota i zrzucenie mu zaopatrzenia oraz powrót do miejsca startu. „Zagrożenia” to radary i działa nieprzyjaciela – można je w uproszczeniu modelować jako prostopadłościany o różnych rozmiarach i wysokości. a) Wykonać symulator środowiska odpowiedzialny za dostarczanie obserwacji agentowi i za śledzenie wykonywanych przez niego akcji. Agent „wyczuwa” radary i broń na kilka kratek wprzód (im wyżej leci tym szybciej może lecieć, ale zużywa wtedy odpowiednio więcej energii). Radary należy omijać w poziomie a wystrzały można ominąć w poziomie albo wznieść się powyżej obserwowanego zasięgu działa. W celu zrzucenia zaopatrzenia dron powinien obniżyć się do minimalnej wysokości Hmin = 0.1 H. Ewentualny kontakt z radarem „unieruchamia” agenta na pewien czas, a trafienie pociskiem zmniejsza jego maksymalną prędkość o połowę. Wynik polega na pomiarze czasu od startu do powrotu przy uwzględnieniu faktu czy zadanie zostało wykonane czy też nie. b) Wykonać bazę wiedzy agenta. Wyrazić wiedzę agenta w rachunku sytuacyjnym języka predykatów (można przyjąć własną implementację zapisu formuł). Zrealizować funkcje TELL i ASK dla komunikacji funkcji głównej agenta z bazą wiedzy. c) Zaimplementować ogólny (uniwersalny) algorytm wnioskowania w przód, w celu wnioskowania o nowych faktach i wnioskowania o wyborze akcji. d) Wykonać prosty interfejs graficzny. W osobnym okienku pokazywać wykonane wnioskowania. Parametry N i H mają być zmienne w rozsądnych granicach. e) Umożliwić wykonanie testowania działania programu dla różnych ustawień początkowych świata. Wykonać testowanie i przeanalizować jego wyniki.

### Koncepcja projektu

W realizacji projektu stworzony zostanie system ekspercki wykorzystujący bazę wiedzy zbudowaną za pomocą klauzul Horna (posiadających jeden wniosek oraz mogących zawierać wiele założeń). Klauzule te zostały wybrane ze względu na to posiadaną właściwość:

¬𝑝 ∨ ¬𝑞 ∨ . . .∨ ¬𝑡 ∨ 𝑢 ≡ (𝑝 ∧ 𝑞 ∧ . . . 𝑡) => 𝑢

Aby opisać poszczególne kroki posłużę się rysunkiem z wykładu:



#### Wykład ITO-mgr 1 strona 32

Dla właściwej oceny tego co się dzieje w środowisku agent posiada czujniki. Za ich pomocą możemy poczynić jakieś obserwacje, które z kolei są podstawą do tego, żeby aktualizować stan drona i wiadomości o tym środowisku. Na podstawie tego aktualnego stanu przy wykorzystaniu regułowej bazy wiedzy można wykonać odpowiednie wnioskowanie oraz wykonać jakieś akcje, które zależeć będą od aktualnego stanu środowiska.

Projekt ten rozpatruję w kategorii zmodyfikowanego 3-wymiarowego świata Wumpusa. Agent (dron) startować będzie z pozycji początkowej w terenie NxNxH komórek i będzie poruszał się w różnych kierunkach w celu odnalezienia pilota. Celem agenta będzie odnalezienie pilota i zrzucenie mu zaopatrzenia oraz powrót do miejsca startu. Agent za pomocą swoich czujników będzie wyczuwał zagrożenie (radary i broń) na kilka kratek w przód, aby móc je ominąć i dotrzeć do celu.

Obserwacje i akcje agenta będą następujące:

* Obserwacje: [ zagrożenie, unieruchomienie, zmniejszenie prędkości, dostosowanie wysokości]

- Agent wyczuwać będzie zagrożenie na kilka kratek w przód w każdym kierunku (będzie to uzależnione od podanego parametru)

- W przypadku gdy agentowi nie uda się ominąć radaru i jego tor ruchu będzie przechodzić przez kratkę z radarem będzie on unieruchomiony

- Gdy agent zostanie trafiony strzałą odczuwa zmniejszenie prędkości

- Gdy agent znajdzie się nad pilotem sprawdza swoją wysokość i jeśli nie jest na odpowiedniej wysokości to wykonuje korektę wysokości do poziomu 0.1H

* Akcje: { RuchWPrzód, RuchWTył RuchWLewo, RuchWPrawo, RuchWGóre, RuchWDół, Upuścić, Zginąć }

- ,,Upuścić” zaopatrzenie, jeśli dron znajdzie się nad pilotem

- ,,Zginąć” agent zginie, jeśli zostanie trafiony przez dwie strzały z rzędu

Wprowadzenie teoretyczne

Dla pełnego wyjaśnienia schematu postępowania w przypadku algorytmu wnioskowania w przód wprowadzę następujące pojęcia:

**Baza wiedzy** składa się z reguł i faktów.

**Reguła** to wyrażenie postaci np. jeśli A to B ( A => B), gdzie A nazywamy **założeniem**, natomiast B(**wnioskiem**/konkluzją)

**Fakty** - są to wyrażenia logiczne, którym przypisano wartość logiczną.

Fakty mogą być **dopytywalne** bądź **niedopytywalne**. Fakty niedopytywalne to są takie, które nie występują jako wniosek.

**Predykat** wskazuje na konkretny obiekt( są to np. założenia, wnioski, fakty).

Dla przykładowej bazy wiedzy:

##### A => B

###### B => C

C jest faktem **niedopytywalnym** bo zawiera wniosek a z kolei B jest faktem **dopytywalnym** bo nie wynika z żadnej innej reguły.

Przy stworzeniu uniwersalnego algorytmu wnioskowania w przód będą wykorzystywane wprowadzone powyżej pojęcia:

* • fakty **niedopytywalne**
* • fakty **dopytywalne**

Dla przykładowej bazy wiedzy:

##### A => D

F, H => G

B => L

D, J => M

C, D => F

A, E => J

Musimy na początku ustalić, które fakty są dopytywalne, a które nie:

|  |  |
| --- | --- |
| **Fakty dopytywalne** | **Fakty niedopytywalne** |
| A | D |
| E | G |
| B | L |
| H | M |
| C | F |
|  | J |

Algorytm będzie po kolei przechodził przez każdą z reguł w bazie wiedzy i pobierał te predykaty, dla których znana jest wartość logiczna i mamy pewność że wartość ta się nigdy nie zmieni.

Warunki, jakie należy spełnić, aby móc zastosować dany algorytm:

###### - Trzeba znać wartości logiczne faktów dopytywalnych

- Będziemy ,,przeskakiwać” po kolejnych regułach (w przypadku braku pobrania kolejnego predykatu wartość pobrana z poprzedniej reguły przechodzi do kolejnej bez zmiany)

- Warunek stopu (pętla do while w której warunkiem jest to że nie dopiszemy żadnego predykatu po sprawdzeniu wszystkich reguł sprawdzając je kolejno w bazie wiedzy ).

###### Iteracja przejścia przez algorytm dla przykładowej bazy danych

|  |  |
| --- | --- |
| Krok | Wartość pobrana |
| A -> D | D |
| F, H -> G | (brak) a zatem zostaje D |
| B -> L | D L |
| D, J -> M | D L |
| C, D -> F | D L F |
| A, E -> J | D L F J |
| --------------------------------------------Kolejna iteracja----------------------------------------------- | |
| A -> D | D L F J |
| F, H -> G | D L F J G |
| B -> L | D L F J G |
| D, J -> M | D L F J G M |
| C, D -> F | D L F J G M |
| A, E -> J | D L F J G M |

###### Wnioskowanie wymaga 2 założeń

1. założenie zamkniętego świata (baza wiedzy, która zostanie stworzona nie będzie dynamiczna (nie będzie można dodać reguł podczas wnioskowania))

2. zasada poprawnego wnioskowania (reguła A =>B (jest spełniona dla par A,B))

|  |  |
| --- | --- |
| A | B |
| 1 | 1 |
| 0 | 1 |
| 0 | 0 |

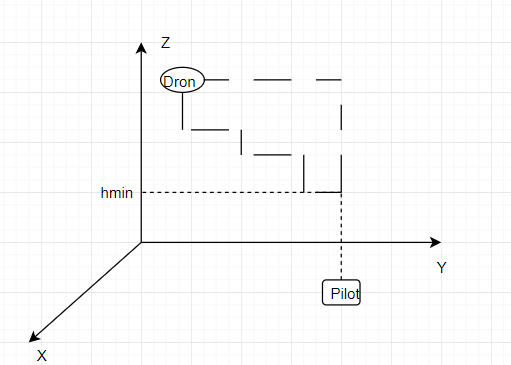
Przy założeniu dwóch wymagań jednocześnie (1 i 2) pozostaną nam pary **A B**

|  |  |
| --- | --- |
| A | B |
| 1 | 1 |
| 0 | 0 |

Wynika to z teorii wnioskowania. Na jedno zapytanie musimy otrzymać jedną odpowiedź.

Początkowo algorytm wnioskowania w przód będzie starał się znaleźć jak najkrótszą ścieżkę.

Poniżej przedstawiono przykładową sytuację:

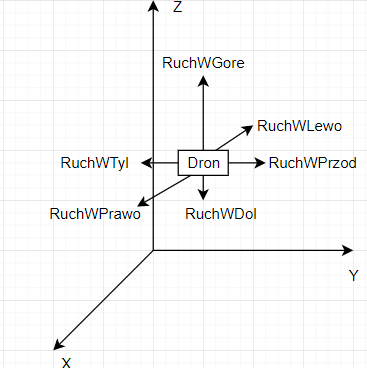


Liniami przerywanymi prowadzącymi od pilota został zaznaczony poziom hmin z którego możliwe jest zrzucenie zaopatrzenia przez agenta. Linie przerywane prowadzące od drona to przykładowe ruchy drona, które spowodują że znajdzie się on na określonym poziomie hmin. Istnieją co najmniej dwa sposoby podejścia do wyznaczenia najkrótszej ścieżki:

1. Agent(dron) dąży do tego aby znaleźć się nad dronem i wtedy zmniejsza swoją wysokość, aby zrzucić mu zaopatrzenie
2. Dron najpierw zniży się do poziomu hmin = 0.1H a następnie będzie próbował dostać się do pilota

Przez ścieżkę będę oznaczał sumaryczną liczbę ruchów jakie musi wykonać dron, aby przedostać się ze swojego położenia początkowego do pilota. W realizacji projektu wybrano 1 podejście. W przypadku gdy na najkrótszej ścieżce agent napotka jakieś przeszkody (radar, broń) ścieżka ta będzie modyfikowana z wykorzystaniem rachunku sytuacyjnego tak, aby uniknąć zagrożeń.

**Założenie początkowe**: dron będzie skierowany w kierunku dodatniej osi Y



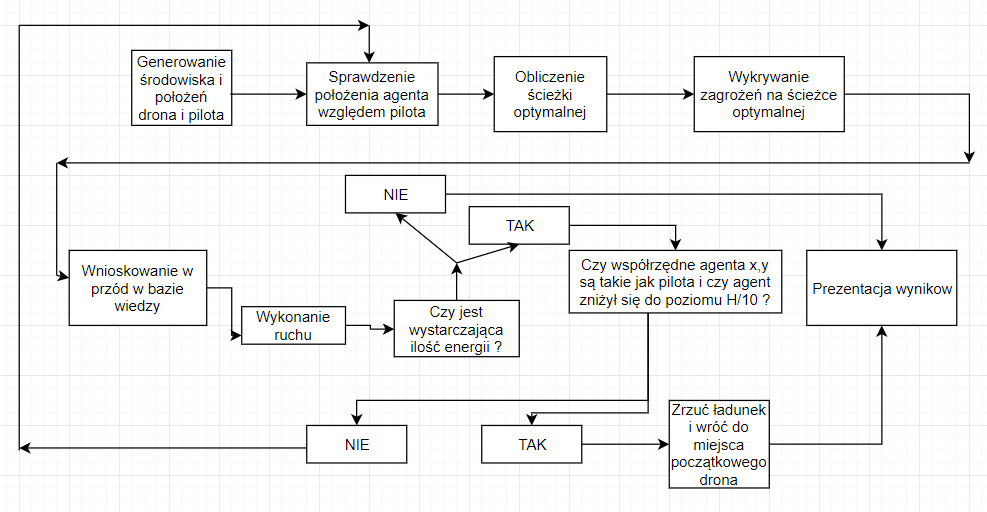
Ruch w kierunku osi X będzie to ruch w prawo lub w lewo

Ruch w kierunku osi Y będzie to ruch w przód lub w tył

Ruch w kierunki osi Z będzie to ruch w górę lub ruch w dół. Oś Z będzie wyznaczać wysokość.

### Ogólna struktura całego programu

Ogólna struktura całego programu wygląda następująco:



**Środowisko** zostało zaimplementowane jako trójwymiarowy wektor o wymiarach NxNxH.

**Predykaty** którymi się posługuję w projekcie zostały zaimplementowane jako struktury posiadające nazwę, wartość logiczna i flagę czy jest ustawiony ( to jest potrzebne w kontekście wykonywania wnioskowania w przód)

**Dron** (agent) został zaimplementowany jako struktura posiadający następujące pola:

- położenie początkowe na osiach x,y,z

- pole if\_pilot\_postition\_achieved (wskazujące czy osiągnięto już pozycję pilota; współrzędna położenia Z = H/10

- pole initial\_speed (jako wartość początkową prędkości drona)

- initial\_energy(jako wartość inicjalną energii)

- sensors\_range (wartość inicjalna zasięgu czujników)

- if\_return\_to\_start (pole sygnalizujące czy dron powrócił do miejsca początkowego)

**Funkcja get\_simulation\_parameters**

Na początku działania programu generowane jest środowisko oraz położenia drona oraz pilota.

Odbywa się to za pomocą funkcji

**get\_simulation\_parameters**(drone, N\_max, N\_min, H\_max, H\_min, initial\_energy\_max, initial\_energy\_min, randomMode)

gdzie zmienne N\_max oraz N\_min, H\_max, H\_min, initial\_energy\_max, initial\_energy\_min są maksymalnymi i minimalnymi wartości N i H określające środowisko a także początkową energię drona.

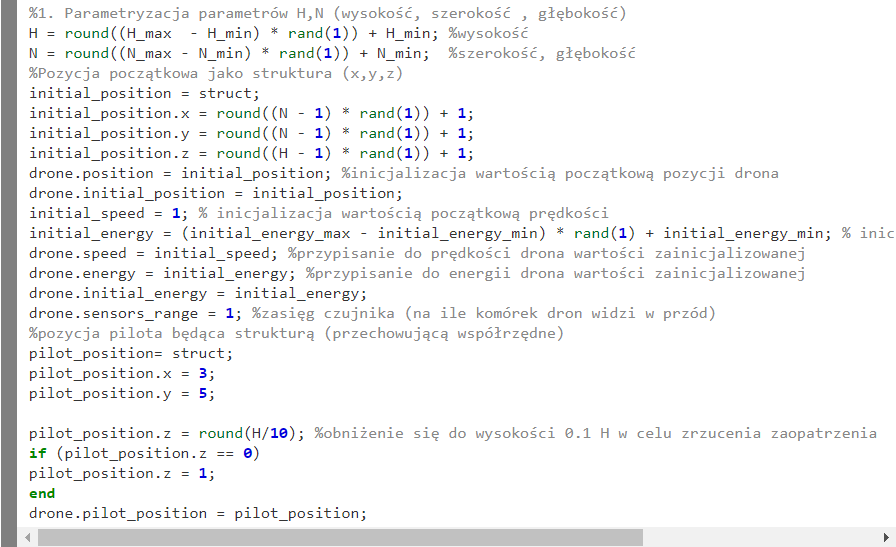
RandomMode zostanie wyjaśnione na etapie GUI.

Zastosowano dwa mechanizmy generowania środowiska:

###### - (1) generowanie za pomocą funkcji losujących

- (2) generowanie poprzez wybranie ręcznie parametrów N i H. (wybrano granice **10-20** dla dobrego zobrazowania co się dzieje w środowisku)

1.Generowanie to odbywa się za pomocą funkcji losujących położenia początkowe drona oraz środowiska; realizowane poprzez wbudowaną funkcję Matlaba **round()**, która służy do zaokrąglania wartości. Realizowane jest to następująco:



Aby zabezpieczyć się przed sytuacją, że wartości początkowego położenia dla jakiejkolwiek z osi wynosiłyby 0 dodano wartość 1 jako najmniejszą wartość jaką może mieć współrzędna.

**Funkcja initialize\_guns\_and\_radars**

Za pomocą funkcji **initialize\_guns\_and\_radars** możliwe jest dodawanie kolejnych przeszkód (radarów i gunów). Zostało to zrealizowane jako prostopadłościany o różnych długościach i wysokościach:



**Główna pętla programu**



opiera się o założenie, że energia drona jest > 0 oraz flaga sygnalizująca czy dron powrócił do miejsca początkowego jest nieustawiona (drone.if\_return\_to\_start=0)

**Funkcja check\_pilot\_presence**



Funkcja ta sprawdza czy dron znajduje się obecnie nad pilotem w pozycji umożliwiającej zrzucenie ładunku

**Funkcja calculate\_optimal\_path**

Za pomocą analizy położenia drona i pilota wyliczana jest ścieżka optymalna (najkrótsza ścieżka łącząca drona i pilota).

Służy do tego funkcja:

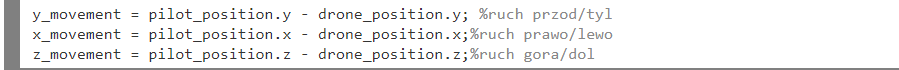


Wyznaczanie ścieżki optymalnej odbywa się na zasadzie wprowadzenia priorytetów ruchu drona (agenta) w zależności od tego jaka jest różnica między położeniami drona i pilota na osiach X,Y,Z.

Ustawianie priorytetów ruchu zależne jest od tego czy została osiągnięta pozycja pilota.

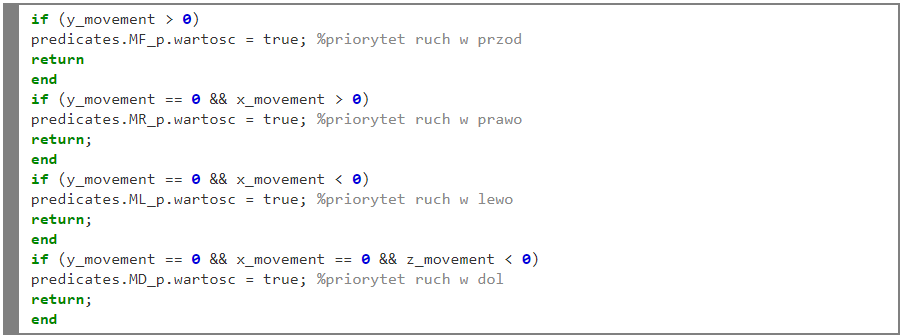
Jeśli nie -> zaczynamy ustawiać priorytety od ruchu w przód

Jeśli tak -> zaczynamy ustawiać priorytet od ruchu w tył



Dla przypadku gdy nie agent nie dotarł jeszcze do pilota i wartość zmiennej y\_movement jest większa od zera mamy do czynienia z ustawieniem priorytetu ruchu w przód, w przeciwnym przypadku będzie to ustawienie priorytetu ruchu w tył.

Przykładowy zestaw ustawienia priorytetów:

****Priorytety ustawione są w następującej kolejności:

* Ruch na osi Y (przód, tył)
* Ruch na osi X (prawo, lewo)
* Ruch na osi Z (góra, dół)

**Funkcja detect\_danger**

Unikanie zagrożeń będzie odbywać się również na zasadzie wyznaczania predykatów ruchu w zależności od tego w którym miejscu patrząc od strony agenta znajduje się zagrożenie.

Odpowiedzialna za tą operację będzie funkcja:



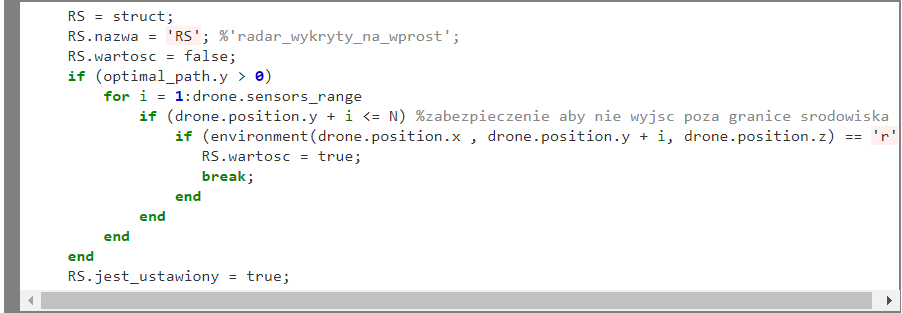
W przypadku gdy dron jeszcze nie zrzucił ładunku pilotowi zaimplementowano funkcję **detect\_danger\_forward\_orientation.**

**Funkcja detect\_danger\_forward\_orientation**



Do wykrycia zagrożenia konieczna będzie informacja o dronie, środowisku, ścieżce optymalnej oraz o predykatach, które będą w tej funkcji aktualizowane w oparciu o obserwacje.

Dla przykładu wykrycia drona w kierunku w przód realizowane jest to następująco:



W przypadku gdy na optymalnej ścieżce agent wykryje radar (oznaczono jako ‘r’) ustawi wartość predykatu **RS**(Radar Straight - czyli radar wykryty na wprost) na wartość True.

W przypadku gdy dron zrzucił już ładunek pilotowi zaimplementowano funkcję **detect\_danger\_backward\_orientation**, która to funkcja jest analogiczna do funkcji detect\_danger\_forward\_orientation z jedyną różnicą taką że ruch w przód obliczony za pomocą funkcji calculate\_optimal\_path jest traktowany jako ruch w tył

To **optimal\_path** (ścieżka optymalna) decyduje o tym jakie predykaty sprawdzające zagrożenie są w obecnym momencie sprawdzane. Jeżeli optymalna ścieżka nie prowadzi na wprost to nie dokonujemy sprawdzania czy radar znajduje się na wprost czy nie.

**Baza wiedzy**

Do zapisu reguł w bazie wiedzy, które zostały rozróżnione na przesłanki i konkluzje użyto następujących skrótów:

**POP** – position over pilot (predykat oznaczający osiągnięcie wysokości pilota)

**DL** – drop load (predykat oznaczający zrzucenie ładunku)

**MF** – move forward (ruch w przod)

**RS** – radar straight (wykryto radar na wprost)

**MF\_p** – move forward priority (ustawiony jest priorytet do ruchu w przód)

**MB** – move backwards (ruch w tył)

**RR** – radar right (wykryto radar z prawej strony)

**RL** – radar left (wykryto radar z lewej strony)

**RU** – radar up (wykryto radar na górze)

**RD** – radar down (wykryto radar na dole)

**MR\_p** – move right priority (ustawiony jest priorytet do ruchu w prawo)

**MR** – move right (ruch w prawo)

**ML** – move left (ruch w lewo)

**MD** – move down (ruch w dol)

**MU** – move up (ruch w gore)

**ML\_p** – move left priority (ustawiony jest priorytet do ruchu w lewo)

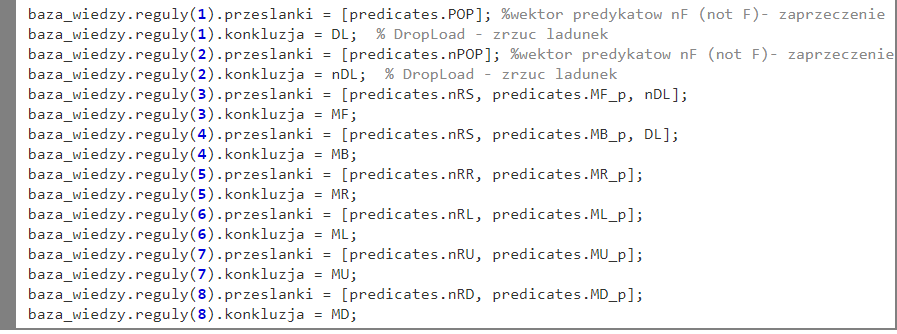
**MD\_p** – move down priority (ustawiony jest priorytet do ruchu w dół)

**MU\_p** – move up priority (ustawiony jest priorytet do ruchu w górę)

**MB\_p** – move backwards priority (ustawiony jest priorytet do ruchu w tył)

Znajdują się także ich negacje poprzedzone przez **n**

Poniżej przedstawiono część reguł z bazy wiedzy:



Dla przykładowej reguły w bazie wiedzy:



Dla powyższego kodu należy to interpretować następująco:

Gdy nie wykryto radaru na wprost oraz predykat priorytetem jest ruch w przód jest ustawiony i nie zrzucono ładunku to wtedy idź w przód.

**Aksjomaty w bazie wiedzy**

Aksjomatami w bazie wiedzy będzie zestaw następujących sytuacji:

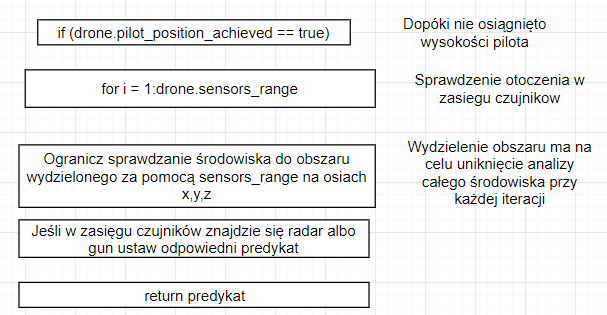
- Położenie początkowe guna i radaru nie zmienia się w trakcie symulacji (po przejściu do kolejnego kroku są w tym samym miejscu)

- Jeśli agent wejdzie w obszar oddziaływania radaru to zostanie on zamrożony na 3 kolejki

- Jeśli agent wejdzie w obszar oddziaływania guna to jego prędkość zostanie zmniejszona o połowę, pod warunkiem że **prędkość** >= 1; wynika to z rozdzielczości zmiany prędkości, która została ustawiona na wartość **1**

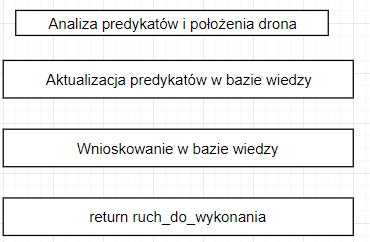
**Funkcja TELL**

Funkcję **TELL (funkcja detect\_danger)** reprezentuje funkcja odpowiedzialna za pobranie aktualnych wartości predykatów oraz ich aktualizacje w podanej powyżej bazie wiedzy:



**Funkcja ASK**

Funkcję **ASK** (**inference**) reprezentuje funkcja odpowiedzialna za pobranie aktualnych predykatów oraz położenia drona i poprzez wnioskowanie reguł w bazie wiedzy ustala kolejny ruch do wykonania przez agenta:



Realizowane jest to w funkcji **inference**



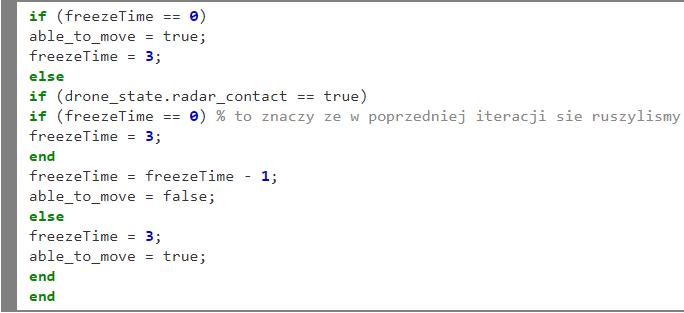
Ruch\_do\_wykonania (**move\_to\_make**)jest to struktura zawierająca pola (x,y,z) i to jest binarne wskazanie w którym kierunku mamy się przemieścić.

**Funkcja check\_drone\_state**

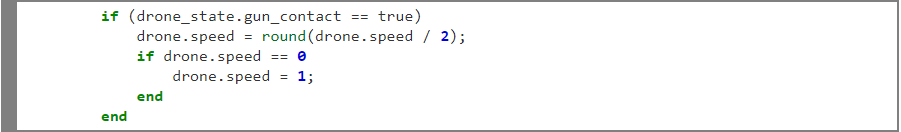
Po wykonanym ruchu następuje sprawdzenie czy agent nie znalazł się w obszarze oddziaływania radaru/guna



W momencie wyjścia z funkcji odpowiedzialnej za analizę bazy wiedzy sprawdzany jest jego otoczenie po wykonaniu jednego ruchu. W sytuacji gdy dron napotka po którejś ze stron (prawo, lewo, góra, dół, przód) jakąś przeszkodę to za pomocą reguł w bazie wiedzy próbuje ją ominąć. W przypadku gdy nie może tego zrobić kontakt z radarem powoduje zamrożenie jego ruchu na 3 kolejki (odpowiada za to flaga **freezeTime**)

****

natomiast kontakt z gunem powoduje zmniejszenie jego prędkości o połowę



Użycie zapisu

****

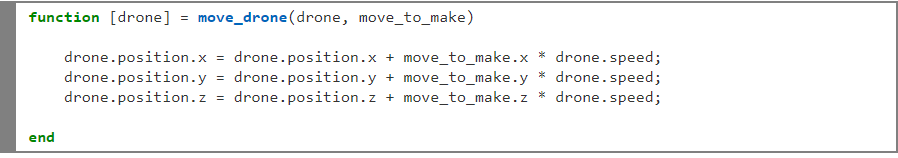
Zabezpiecza, przed sytuacją kiedy prędkość drona byłaby 0( żeby agent nie stał w miejscu).

Dzięki zmiennej **able\_to\_move** możemy stwierdzić czy jest możliwe wykonanie ruchu analizując aktualne położenie drona.

**Move\_to\_make** pokazuje tylko kierunek w który należy się przemieścić (jest to wynik wnioskowania z bazy wiedzy), natomiast w zależności od prędkości będziemy się poruszać za pomocą funkcji move\_drone

**Funkcja move\_drone**

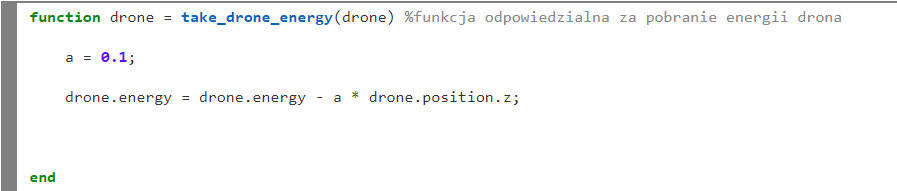
W przypadku, gdy możliwe jest wykonanie ruchu wykonywana jest funkcja move\_drone:



Pozycje drona w osiach x,y,z są uaktualniane w oparciu o dotychczasowe położenie oraz ruch do wykonania uwzględniając aktualną prędkość

**Funkcja take\_drone\_energy**

Po wykonaniu każdego ruchu wartość energii drona jest zmniejszana za pomocą funkcji **take\_drone\_energy**

****

A jest współczynnikiem. Przyjęto **a** = 0.1 aby energia drona zbyt szybko nie zmniejszała się.

Pobierane energia drona jest funkcją wysokości ponieważ modelujemy w ten sposób taką sytuację że dron może szybciej lecieć, jednak odbywa się to kosztem energii.

**Funkcja append\_move\_to\_message**

Za pomocą funkcji append\_move\_to\_message przekazywane są parametry ruchu ze struktury moves\_and\_states:



Struktura ta jest odpowiedzialna za zbieranie ruchów a każdej iteracji symulacji jakie wykonał dron



Parametry te przekazywane są do bufora message, który to bufor wyświetla informacje na temat ruchu w GUI

**Funkcja check\_finish\_conditions**



Funkcja ta służy do sprawdzenia czy dron powrócił do miejsca startu

**Funkcja plot\_simulation**

W momencie gdy program wyjdzie z głównej pętli wyświetlane są wyniki za pomocą funkcji



**,** która to funkcja przy pomocy zbieranych wyników w każdej iteracji wyświetla aktualne środowisko, położenie drona oraz pilota, rozmieszczenie przeszkód a także drogę jaką pokonał dron od miejsca początkowego do pilota.

**Funkcja analyze\_results**

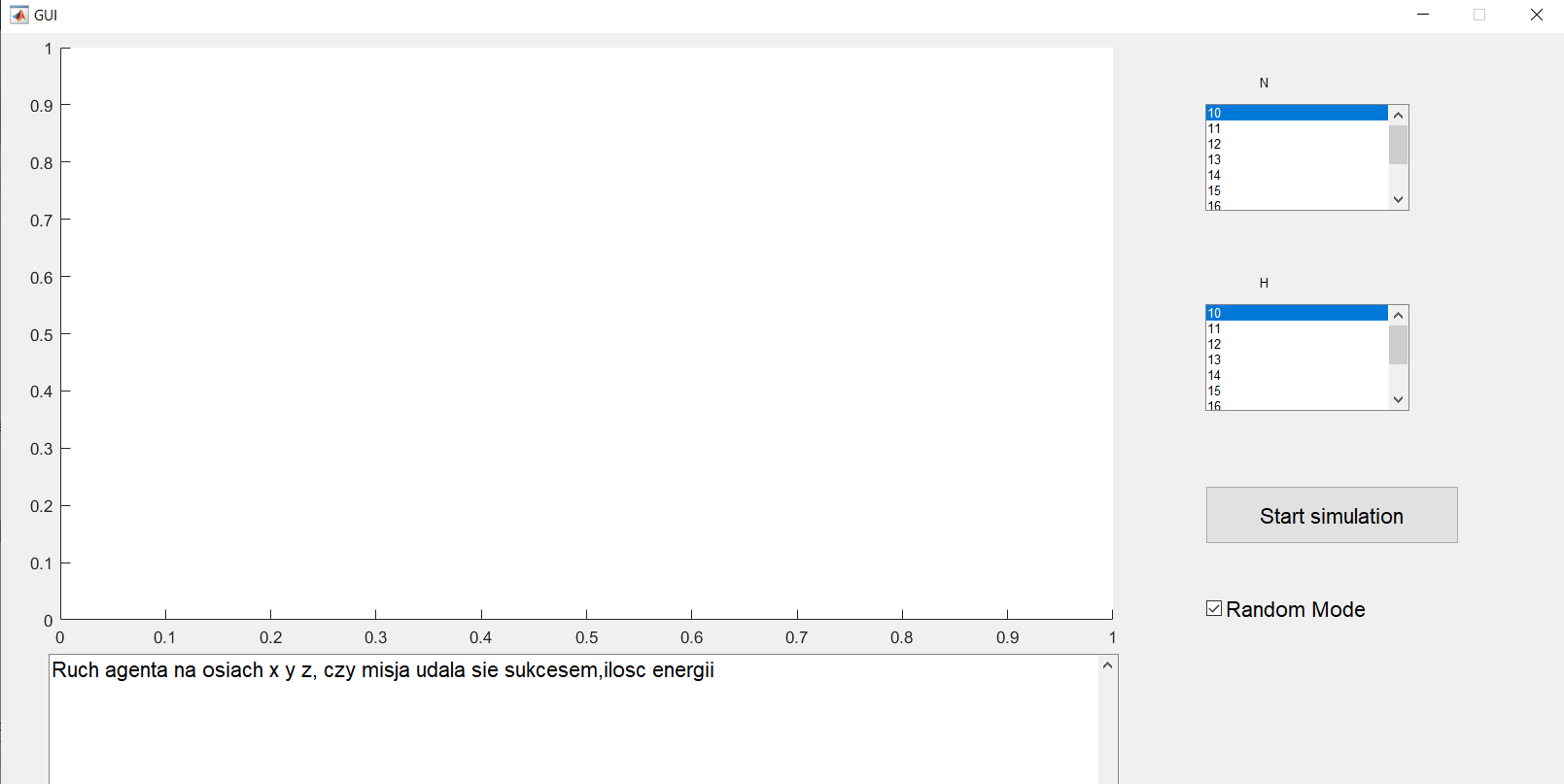
Zbieraniem wyników dotyczących analizy ruchu od drona do pilota oraz informacji o tym jakie są inicjalne wartości energii zajmuje się funkcja **analyze\_results**.



Umożliwia ona też prezentację wyników wnioskowania. Wyświetla ona średnią prędkość, informację o tym czy misja się powiodła, czas trwania symulacji oraz ruchy jakie wykonał agent na poszczególnych osiach x,y,z aby dostać się z miejsca startowego do pilota oraz z powrotem( pod warunkiem że misja zakończy się sukcesem)

### **Główne okno programu**:

W ramach wymagań projektowych zrealizowany został graficzny interfejs użytkownika:



Po prawej stronie znajdują się parametry **N** i **H** pozwalające na wybór wartości N i H w środowisku przed jego wygenerowaniem. Możliwy wybór to zakres 10-20 dla dobrego zobrazowania środowiska, drona, pilota oraz przeszkód w postaci radarów i gunów.

Checkbox **Random** **Mode** jest wstawiony dla zobrazowania dwóch możliwości testowania programu, za pomocą wbudowanych funkcji round() co jest możliwe w przypadku, gdy checkbox jest zaznaczony.

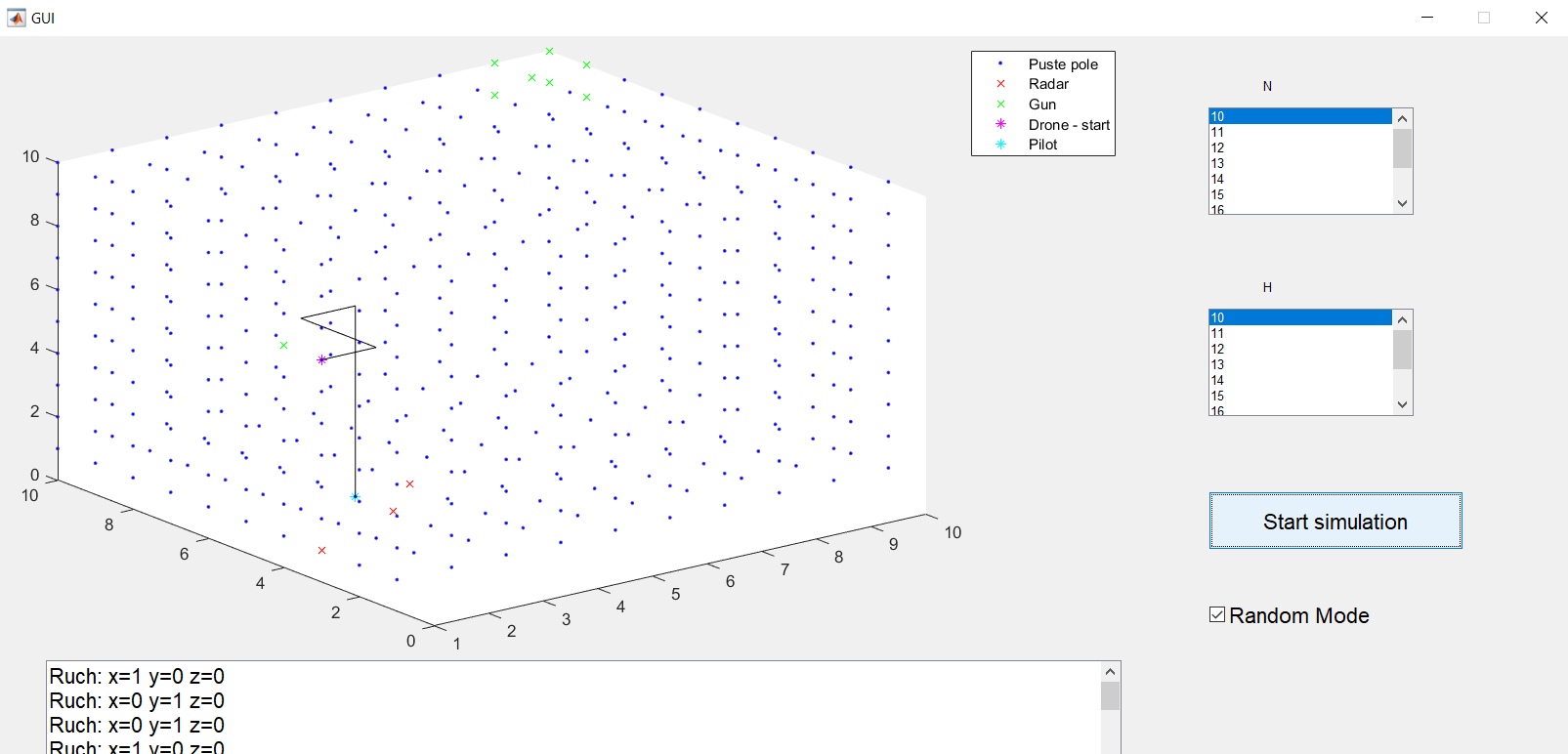
W przypadku braku zaznaczenia checkboxa **Random** **Mode** położenia początkowe będą mogły zostać manualnie ustawione przez użytkownika w skrypcie **get\_simulation\_parameters.m**

Przycisk **Start** **simulation** powoduje start symulacji z ustawionym środowiskiem.

Na wskazanym wyżej wykresie pojawiać się będzie środowisko, położenie drona, pilota, droga jaką przebył agent w celu dotarcia do pilota a także rozmieszczenie przeszkód.

Poniżej umieszczony będzie dokładny ruch agenta w kierunku osi X, Y, Z jakie od punktu startowego wykonał aby dojść do pilota oraz z powrotem do miejsca początkowego.

Dla przykładowego przebiegu po wciśnięciu **Start** **Simulation** wraz z zaznaczonym checkboxem **Random** **Mode**:



Jak widać na załączonym obrazku dostępna jest także legenda wykresu, dzięki czemu można w łatwy sposób sprawdzić położenie poszczególnych elementów.

Możliwe jest również rotowanie wykresu, dla pełnego zobrazowania poszczególnych położeń

Gdzie cała informacja poniżej zawiera poniższe informacje:

Ruch: x=1 y=0 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=1 y=0 z=0

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=-1 y=0 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=-1 y=0 z=0

Random Mode ON

N=10 H=10

Misja zakończona powodzeniem

Symulacja trwała: 21 iteracji

Średnia prędkość: 0.95238

Energia początkowa: 39.6722 Pozostało energii: 29.2722

Scenariusze testowe:

1. Pilot jest na granicy środowiska i sprawdzenie czy dron nie wyjdzie poza granice środowiska

Drone.initial.x=9

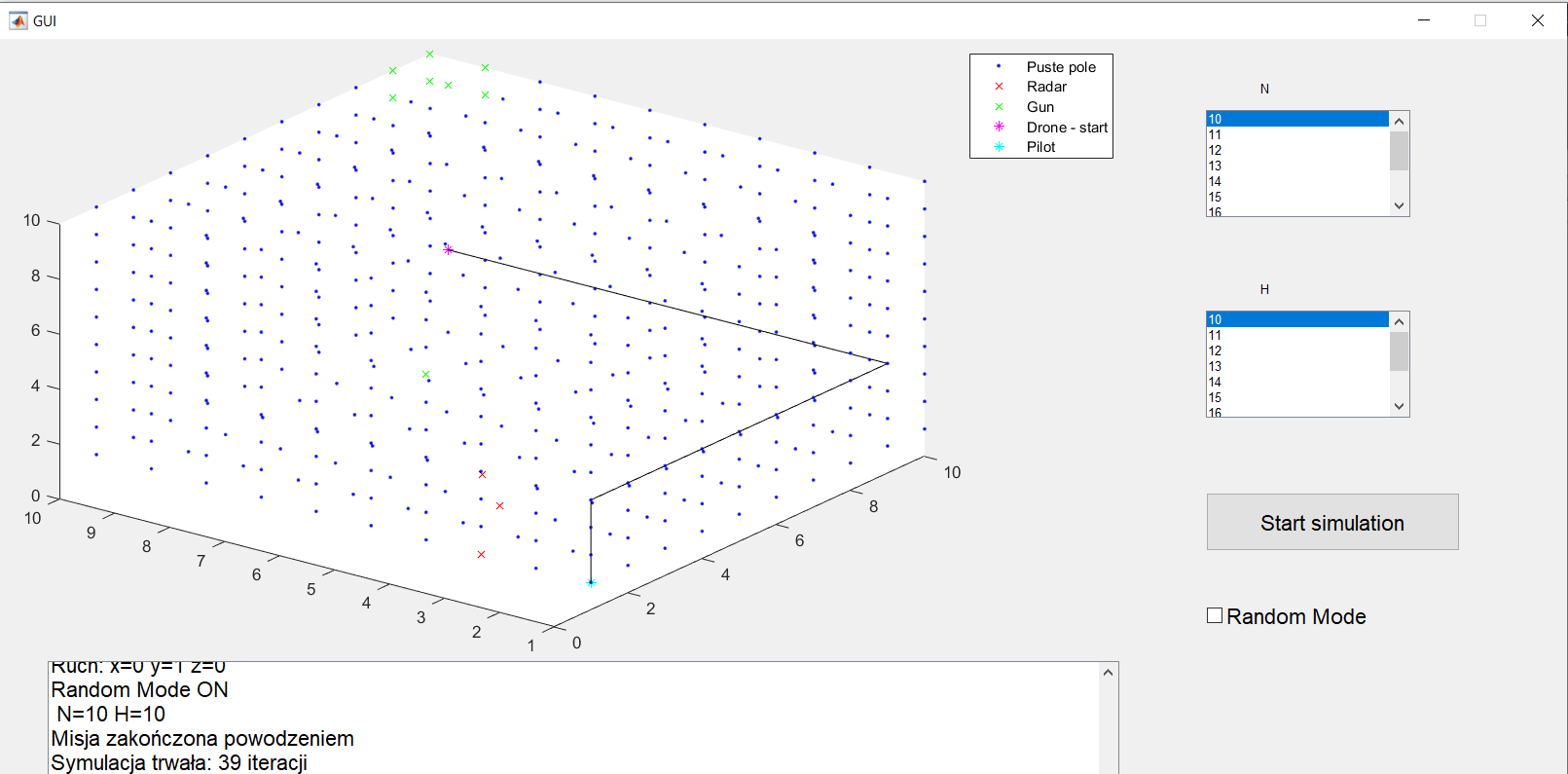
Drone.initial.y=9

Drone.initial.z=4

Pilot.position.x= 1

Pilot.position.y= 1

Pilot.position.z = 1



1. Środowisko jest pozbawione przeszkód (radarów i gunów)

Drone.initial.x=9

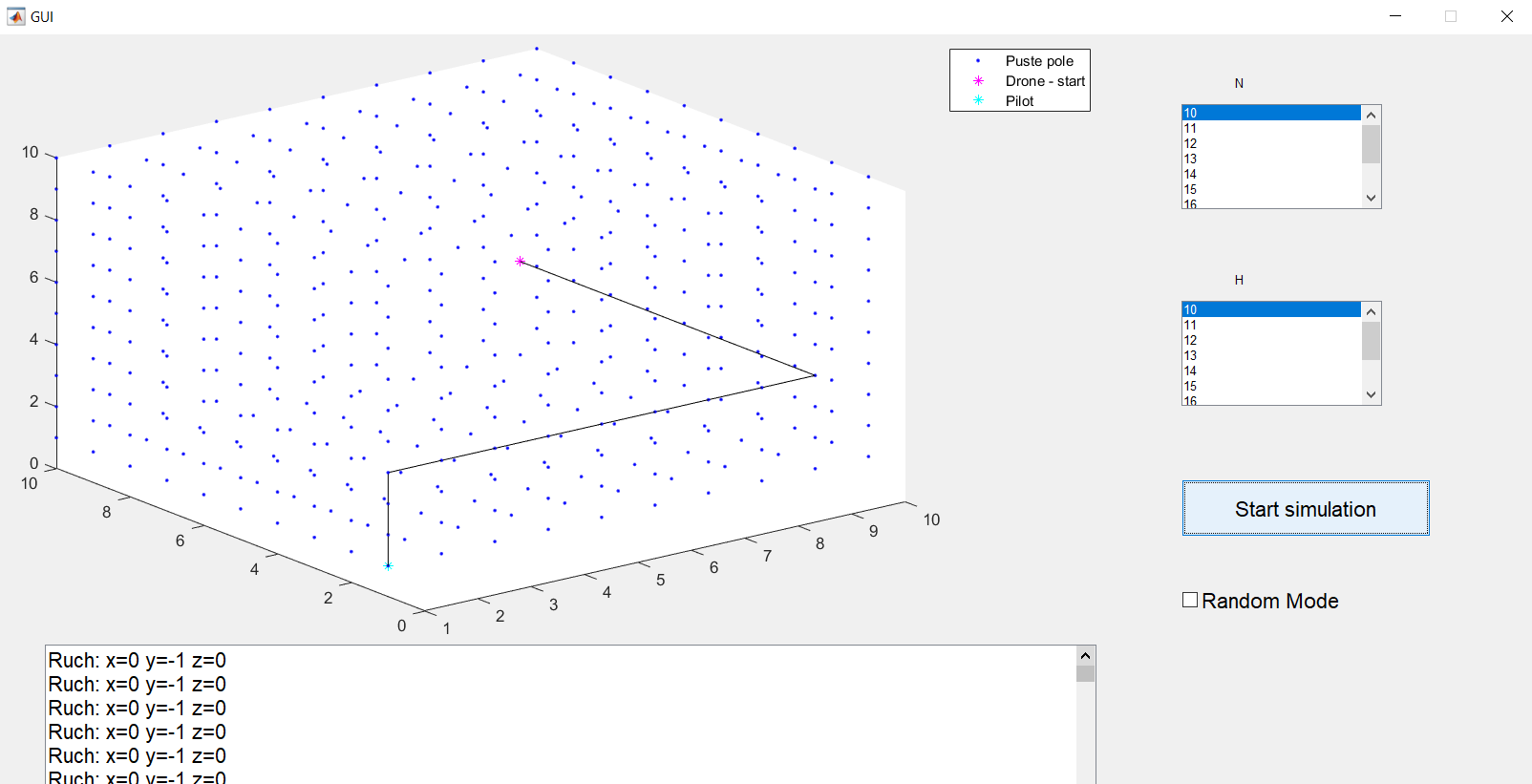
Drone.initial.y=9

Drone.initial.z=4

Pilot.position.x= 1

Pilot.position.y= 1

Pilot.position.z = 1



Cała informacja:

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Random Mode ON

N=10 H=10

Misja zakończona powodzeniem

Symulacja trwała: 39 iteracji

Średnia prędkość: 0.97436

Energia początkowa: 25.6062 Pozostało energii: 11.3062

1. Pilot jest wyżej niż dron ( z pilota > z drona)

Drone.initial.x=9

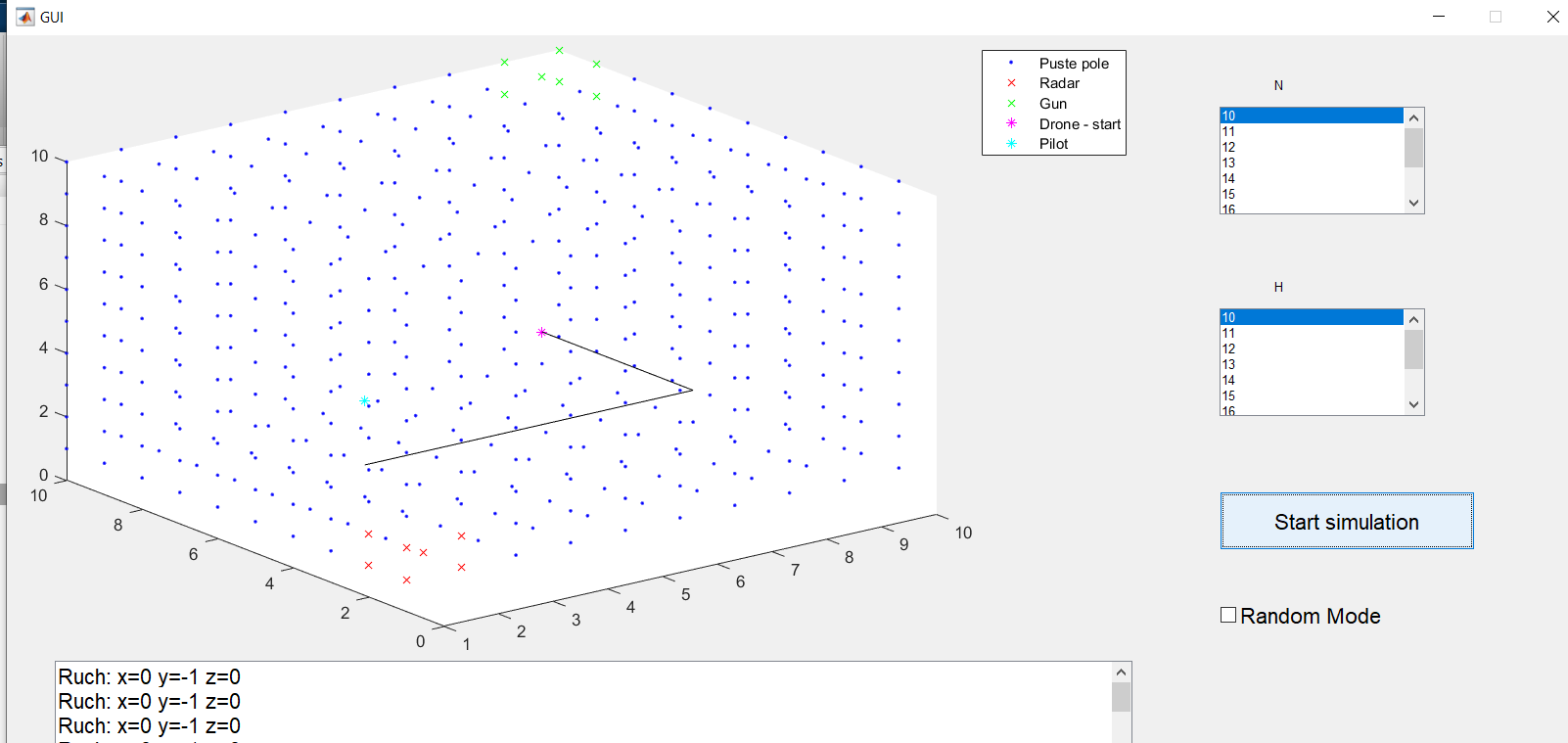
Drone.initial.y=9

Drone.initial.z=2

Pilot.position.x= 3

Pilot.position.y= 5

Pilot.position.z = 4



Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Random Mode OFF

Misja zakończona powodzeniem

Symulacja trwała: 21 iteracji

Średnia prędkość: 0.95238

Energia początkowa: 27.5325 Pozostało energii: 23.5325

Czy tak powinno być ??

1. Dron jest przed pilotem (y pilot > y dron)

Drone.initial.x=9

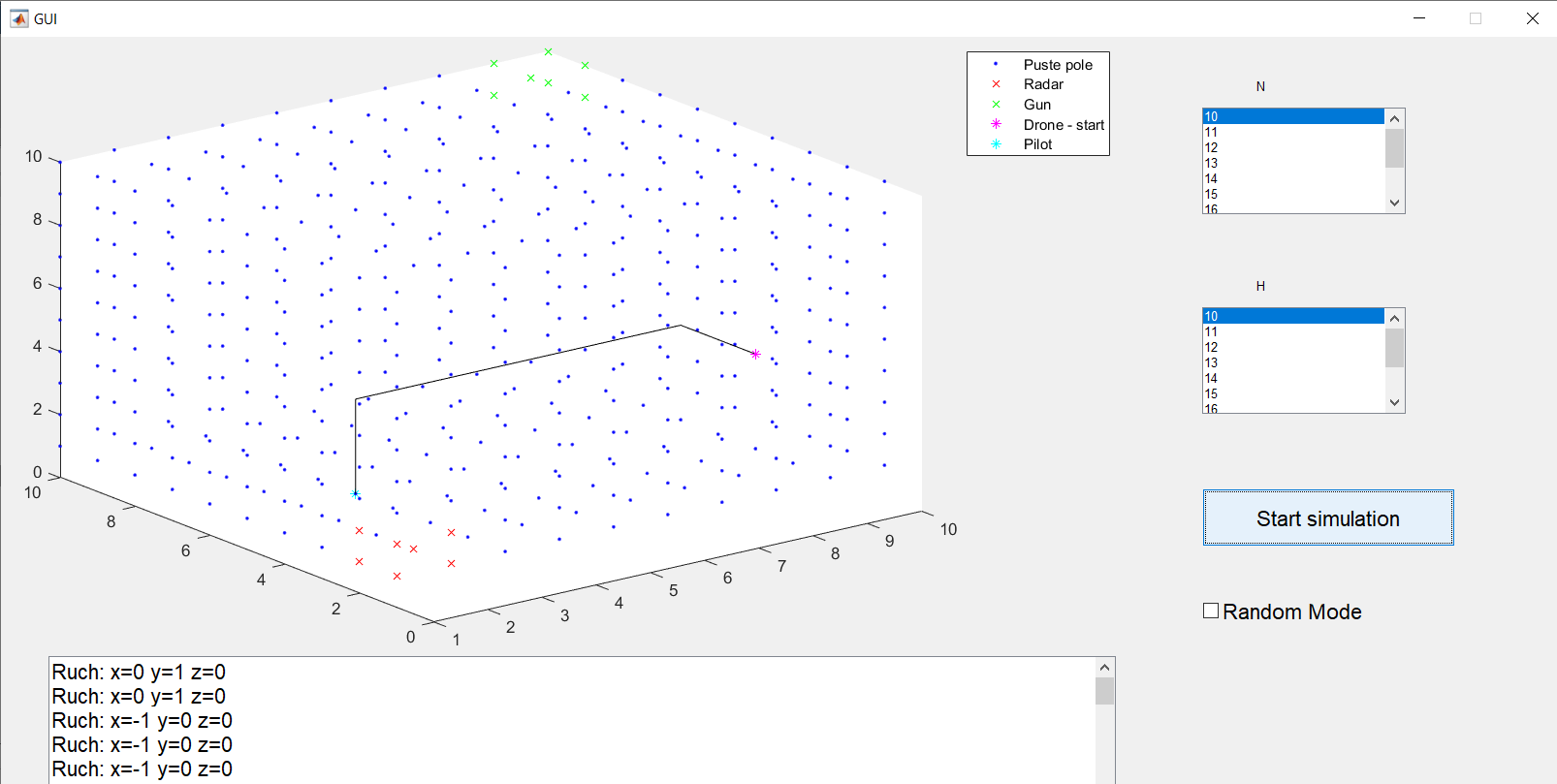
Drone.initial.y=3

Drone.initial.z=4

Pilot.position.x= 3

Pilot.position.y= 5

Pilot.position.z = 1



Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Random Mode OFF

Misja zakończona powodzeniem

Symulacja trwała: 23 iteracji

Średnia prędkość: 0.95652

Energia początkowa: 30.5498 Pozostało energii: 22.6498

1. Dron jest za pilotem (y pilot < y dron)

Drone.initial.x=9

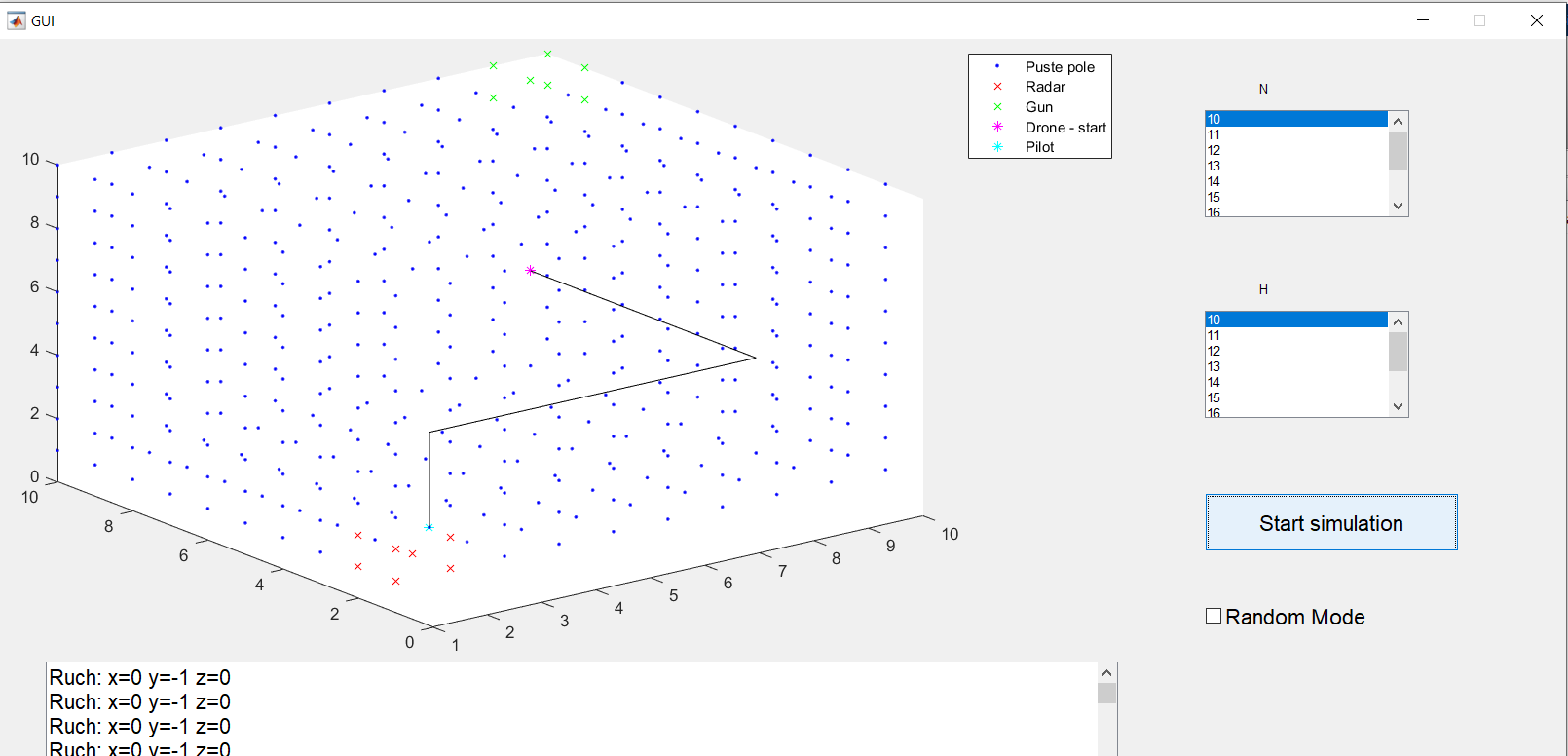
Drone.initial.y=9

Drone.initial.z=4

Pilot.position.x= 3

Pilot.position.y= 3

Pilot.position.z = 1



Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=0 y=-1 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=-1 y=0 z=0

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=-1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=0 y=0 z=1

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=1 y=0 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Ruch: x=0 y=1 z=0

Random Mode OFF

Misja zakończona powodzeniem

Symulacja trwała: 31 iteracji

Średnia prędkość: 0.96774

Energia początkowa: 44.9249 Pozostało energii: 33.8249

1. Wsadzić pilota pomiędzy radar i gun (dostęp do pozycji pilota wymaga przejścia przez jakieś przeszkody)

environment(3:6, 4:6, 5:6) = 'r'; %(od 3 do 5, od 4 do 6, od 5 do 7) %wstawienie do pól konkretnych zakresów literki 'r',która będzie oznaczać, że w tych miejscach znajdować się będzie radar

environment(1:2, 1:2, 1:2) = 'r';

environment(9:10, 9:10, 9:10) = 'g';

environment(8:9, 8:9, 8:9) = 'g';

Drone.initial.x=9

Drone.initial.y=9

Drone.initial.z=8

Pilot.position.x= 7

Pilot.position.y= 7

Pilot.position.z = 7

1. Na całej szerokości środowiska rzucamy guna
2. Na całej szerokości środowiska rzucamy radar
3. Testowanie zależności dla różnych wartosci energii i zbadać wpływ na testowanie
4. Testowanie zaleznosci dla roznych N i H i zbadac wpływ na testowanie

Wnioski:

Dla przykładów testowych nr …. Działa dobrze przy założeniu odpowiednio dużej ilości energii (pilot będzie oddalony od drona o 50 jednostek np.) a energia będzie np. 30

Algorytm może wykonywać te same ruchy cały czas (np. prawo, lewo) (winą jest ustawienie środowiska, bo w innym przypadku dron narazi się na kontakt z przeszkodą, bądź takie specyficzne ustawienie przeszkód że algorytm jest zbudowany w oparciu o zamknięty zestaw reguł -> w takim przypadku należałoby przeanalizować daną sytuację i dodać odpowiednią regułę w bazie wiedzy.

Agent może nie ruszać się przez dłuższy czas ze względu na specyficzną sytuację w której agent jest zastawiony z każdej strony radarami

Przy znacznym zwiększeniu rozmiarów środowiska macierz trójwymiarowa staje się nieoptymalnym rozwiązaniem -> wyjściem z takiej sytuacji byłoby optymalizacja kodu pod kątem wydajnościowym i pamięciowym.

Projekt był rozwijany w oparciu o system kontroli wersji GiT