ARR 2016

Projekt przejściowy — Symulator laboratorium L1.5

24 stycznia 2017

Spis treści

1. Wstęp . . . 5
1.1. Motywacja . . . 5
1.2. Cel projektu . . . 5
2. Wykorzystane narzędzia . . . 7
2.1. Zarządzanie projektem . . . 7
2.1.1. Git . . . 7
2.1.2. Trello . . . 7
2.2. Docker . . . 7
2.3. Gazebo . . . 7
2.4. ROS . . . 7
2.5. Qt . . . 7
2.6. Inventor . . . 7
3. Opis systemu . . . 9
3.1. Okienka . . . 9
3.1.1. Konfiguracja świata . . . 9
3.1.2. Zarządzanie robotami . . . 9
3.1.3. Wyniki symulacji . . . 10
3.2. Pluginy . . . 10
3.2.1. GUI . . . 10
3.2.2. World . . . 10
3.3. ROS + catkin . . . 10
3.4. Modele . . . 10
4. Testy . . . 11
4.1. Przygotowanie . . . 11
4.2. Obserwacje . . . 11
5. Wnioski . . . 13
5.1. Serwer . . . 13
Bibliografia . . . 15

1. Wstęp

Środowisko do symulowania działania robotów w warunkach laboratorium L1.5 został wykonany na zajęciach projektu przejściowego grupy ARR. Środowisko jest dostarczony w postaci kontenera dockera na którym znajduje się system operacyjny ubuntu 16.04 ros kinetic oraz gazebo 7.5.

Celem projektu było umożliwienie przygotowania się studentom do zajęć laboratoryjnych poprzez testowanie napisanych przez siebie programów w środowisku zbliżonym do udostępnianej przestrzeni podczas zajęć praktycznych. Dostarczony projekt umożliwia wybór laboratorium i robotów pionier oraz pozwala na dodawanie elementów sceny takich jak przeszkody. Z wykorzystaniem środowiska ros można sterować robotami. Symulator pozwala również na zmianę pozycji robota.

Osoba której zadanie ograniczać się będzie do wykonania ćwiczeń laboratoryjnych powinna zapoznać się z instrukcją z dodatku A. W celu ułatwienia rozwoju i modyfikacji projektu stworzony został dodatek B.

1.1. Motywacja

1.2. Cel projektu

2. Wykorzystane narzędzia

2.1. Zarządzanie projektem

2.1.1. Git

2.1.2. Trello

2.2. Docker

2.3. Gazebo

2.4. ROS

2.5. Qt

2.6. Inventor

3. Opis systemu

Symulator robotów jest doskonałym narzędziem dla każdej osoby zajmującej się robotyką. Pozwala szybko przetestować różne algorytmy i konstrukcje oraz skomplikowane systemy realizujące niecodzienne scenariusze. Jednym z takich narzędzi jest darmowy program Gazebo, przeznaczony do tworzenia dokładnych i efektywnych symulacji robotów działających w złożonych środowiskach. Posiada zaawansowany silnik fizyki, wysokiej jakości grafikę oraz wygodne i programowalne interfejsy.

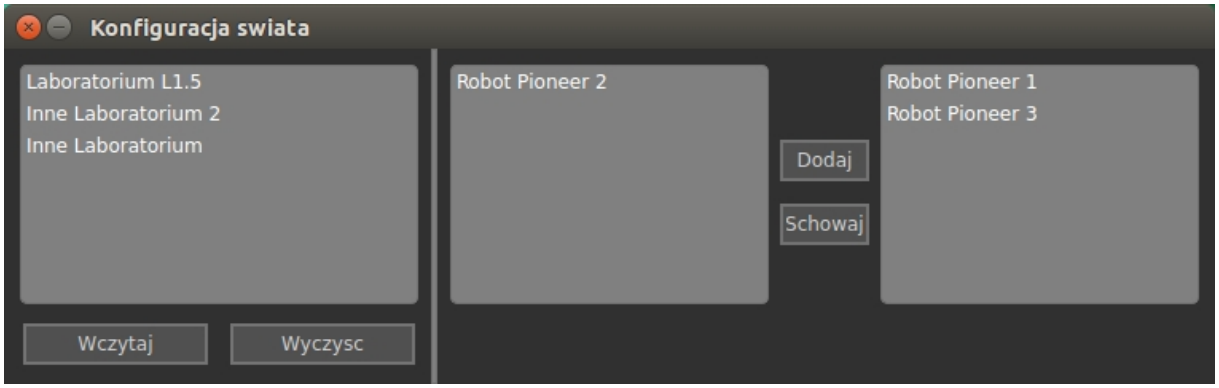
To powyżej to próba tłumaczenia opisu poniżej ze strony Gazebo :D

Robot simulation is an essential tool in every roboticist's toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant community.

3.1. Okienka

Funkcjonalności + Implementacja

3.1.1. Konfiguracja świata



Rysunek 3.1. Okno konfiguracji świata

3.1.2. Zarządzanie robotami

Okienko `Zarządzanie robotami` umożliwia ustawianie pozycji oraz orientacji wybranego robota na scenie. Zakładki umożliwiają wybór *Pioneera*, którego pozycję chcemy

zmienić. Aktywne są tylko zakładki skojarzone z robotami dodanymi aktualnie do świata. W polach odpowiedzialnych za pozycję i orientację robota kolor informuje o poprawności wprowadzonych danych – zielony oznacza poprawnie wypełnione pola, natomiast czerwony błędnie.

Rysunek 3.2. Okno zarządzania robotami

Implementacja

Klasa odpowiadająca za całe okienko to `RobotManagementWindow`. Dziedziczy ona po klasie `QDialog`. Jej kluczowe metody to:

```
public slots:
  void onAddNewRobot(int id);
  void onHideRobot(int id);
```

Odpowiadają one za to co dzieje się w oknie odpowiednio w momencie dodania robota i schowania go. Mianowicie, w momencie otrzymania odpowiednich sygnałów aktywowana bądź dezaktywowana jest stosowna zakładka.

Klasa odpowiadająca za wygląd zakładki to `RobotManagementTab`. Dziedziczy ona po klasie `QWidget`. Jej kluczowe metody to:

```
void receivedMsg(const boost::shared_ptr<const gazebo::msgs::Int> &msg);
private slots:
  void on_pushButtonUstaw_clicked();
  void on_pushButtonReset_clicked();
```

3.1.3. Wyniki symulacji

3.2. Pluginy

3.2.1. GUI

3.2.2. World

3.3. ROS + catkin

3.4. Modele

4. Testy

4.1. Przygotowanie

4.2. Obserwacje

5. Wnioski

5.1. Serwer

Bibliografia

[1] *Gazebo – strona internetowa projektu*. http://www.gazebosim.org/.
[2] *ROS – dokumentacja*. http://wiki.ros.org/.