Systemy zdarzeniowe

Raport

Opracowanie rozproszonego systemu koordnacji agentów mobilnych

Skład grupy:
Paweł Bogner
Marcin Dmochowski
Bartosz Folta
Grzegorz Maj
Krzysztof Nomejko

Prowadzący:
dr inż. E. Roszkowska

4 lutego 2014

1 Zarządzanie projektem

Zdecydowano, że zostanie wykorzystana tradycyjna, płaska struktura zarządzania z jednym liderem (koordynatorem). Do zadań lidera należy podejmowanie krytycznych decyzji projektowych, rozstrzyganie sporów oraz kontrolowanie postępu prac nad przydzielonymi zadaniami.

W kwestii rozstrzygania sporów, strona konfliktu ma prawo do przedstawienia problemu na forum grupy, w celu jego wspólnego przedyskutowania. W świetle przedstawionych argumentów i poglądów lider ma obowiązek podjąć decyzję rozstrzygającą.

Za termin regularnych spotkań przyjęto termin zajęć odbywających się w ramach kursu „Systemy zdarzeniowe". Lider zespołu ma prawo zarządzenia dodatkowego spotkania organizacyjnego na wniosek jednego lub kilku członków zespołu. Osoba wnosząca o zorganizowanie zebrania ma obowiązek wykazać, że spotkanie jest niezbędne w celu dalszego rozwoju projektu.

Do przewidzianych środków komunikacji zdalnej należą „Google groups", „Redmine" oraz rozmowy telefoniczne. W celu składowania i wymiany dokumentów zostanie wykorzystane oprogramowanie „SVN". Każdy z członków grupy na obowiązek korzystać z tego programu. Jako mechanizm monitorowania postępów prac zostanie wykorzystana usługa „Redmine". Jest to narzędzie, które pozwala przydzielić zadanie danemu członkowi lub członkom zespołu, określić ramy czasowe jego wykonania, oraz śledzić postęp prac.

Uznano, że każdy członek grupy zostanie obdarzony prawami własności intelektualnej do części projektu, za której zrealizowanie był odpowiedzialny.

2 Opis logiki

Działania w systemie są odpowiedzią na zdarzenia wynikające z komunikacji serwer-symulator.

1. pobranie informacji początkowych od serwera: współrzędne sektora początkowego,
2. wysłanie zapytania o pozwolenie wjazdu do kolejnego sektora,
3. oczekiwanie na pozwolenie, przy jednoczesnym poruszaniu się zgodnie z wektorem wyznaczonym przez pola potencjałów,
4. otrzymanie pozwolenia na wjazd do kolejnego sektora, ustawienie przeciwnego potencjału na ścianie sąsiadującej z kolejnym sektorem, otrzymanie współrzędnych następnego sektora docelowego,
5. przejazd do kolejnego sektora,

6. zwolnienie poprzedniego sektora, przejście do punktu 2.

3 Zachowanie robotów wewnątrz pola

Zachowanie robotów wewnątrz pola zamodelowane jest za pomocą metody sztucznych pól potencjałów. Potencjały ustalane są w następujący sposób:

- robot oczekujący na zezwolenie wyjazdu z pola widzi wszystkie ściany jako spolaryzowane ładunkiem o znaku zgodnym ze znakiem jego ładunku (rys. 1), a jego wektor sił ma składowe opisane następującymi równaniami:

$$F_x = A * ((X - x)^2 - x^2)$$

$$F_y = A * ((Y - y)^2 - y^2),$$

- robot wykonujący manewr przejazdu do innego pola widzi dodatkowy ładunek na ścianie, w kierunku której ma zmierzać (umieszczony z jej prawej strony, aby zapobiegać kolizjom) (rys. 2), a do składowych jego wektora sił dodaje się człon odpowiedzialny za modelowanie dodatkowego potencjału:

$$F_{xM} = +\frac{B}{(X_p - x)^2}$$

$$F_{yM} = +\frac{B}{(Y_p - y)^2},$$

- dwa roboty znajdujące się w tym samym polu zawsze są spolaryzowane ładunkami punktowymi o jednakowych znakach (rys. 3), a do wektora sił dodawany jest w tym wypadku następujący człon modelujący siłę wzajemnie je odpychającą:

$$F_{xD} = +\frac{C}{\mathrm{sgn}(x - x_D)(x - x_D)^2}$$

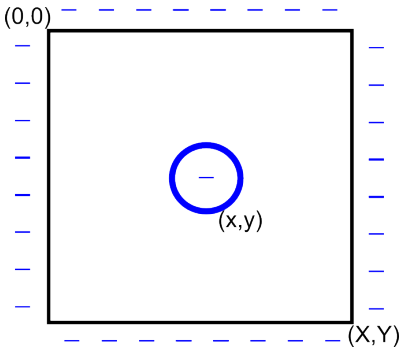
$$F_{yD} = +\frac{C}{\mathrm{sgn}(y - y_D)(y - y_D)^2},$$

- w wypadku, jeśli w polu znajdują się dwa roboty, każdy widzi inną polaryzację ścian – taką, aby zgadzała się ona z jego stanem ruchu (oczekiwanie, zezwolenie na przejazd),

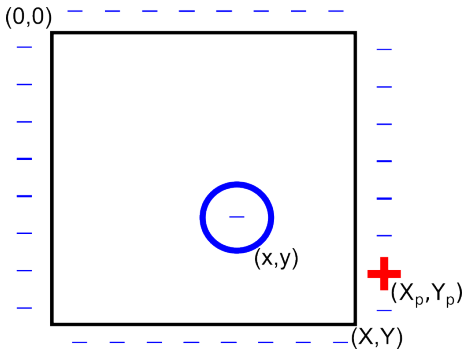
- przykład — kiedy w polu znajdują się dwa roboty i robot, dla którego obliczamy wypadkowy wektor sił, wyjeżdża z pola, składowe jego sił opisane są nastepującymi równaniami:

$$F_x = A((X - x)^2 - x^2) + B(X_p - x)^2 + C(x - x_D)^2$$

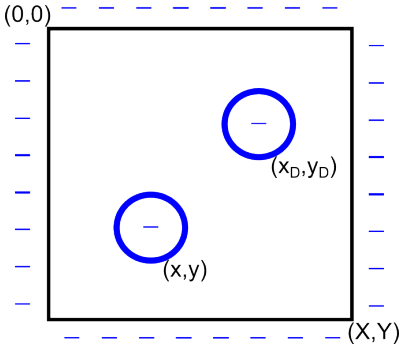
$$F_y = A((Y - y)^2 - y^2) + B(Y_p - y)^2 + C(y - y_D)^2.$$



Rysunek 1: Robot oczekujący na pozwolenie na wyjazd z komórki



Rysunek 2: Robot wyjeżdżający z komórki



Rysunek 3: Dwa roboty oczekujące na pozwolenie na wyjazd z pola

4 Zarys struktur danych

4.1 Scena

W zaproponowanym modelu scena (plansza) jest strukturą, zawierającą tablicę komórek, a także ewentualne informacje związane z modelem robota wspólne dla wszystkich. Pojedyncza komórka zawierałaby informacje o swoich wymiarach oraz listę robotów, które aktualnie się w niej znajdują, a także wskaźnik do struktury zawierającej modele dla metody pól potencjałów.

4.2 Dane wymieniane z serwerem

Serwer ma zadanie planowania tras dla robotów, wobec czego pożądana informacja dla każdego z robotów to następna komórka, do której ten ma się kierować oraz zezwolenie bądź brak zezwolenia na wjazd do niej. W przypadku braku zezwolenia robot zatrzymałby się w bieżącej komórce; w przeciwnym wypadku przejechałby od razu do następnej komórki.

Klient miałby za zadanie wysyłania zapytań o dalsze drogi robotów oraz o pozwolenie na ich realizację, a także informowałby o wykonanych zadaniach i zjeździe robotów ze sceny.

5 Protokół komunikacji z serwerem

Protokół komunikacji jest identyczny z zaimplementowanym w zalążku przez Adama Klamę. Opiera się on na transmisji pakietów danych o odpowiednich nagłówkach poprzez protokół TCP/IP. Wszystkie przekazywane wartości liczbowe mają typ danych int32_t (czterobajtowy *signed integer*).

Przebieg komunikacji z serwerem po uruchomieniu aplikacji:

1. Klient podłącza się do serwera.
2. Klient wysyła do serwera żądanie rejestracji robota:
   - ramka zapytania ma rozmiar 8 bajtów oraz nagłówek oznaczony 1 (`REGISTER_ROBOT`), zawiera ona kolejno lokalne id robota (id w kliencie) oraz promień robota;
   - ramka odpowiedzi ma rozmiar 24 bajty oraz nagłówek oznaczony 2 (`REGISTER_ROBOT_RESP`), zawiera ona kolejno globalne id robota (w serwerze), lokalne id robota (w kliencie), rozmiar pojedynczej komórki planszy w osi X, rozmiar pojedynczej komórki planszy w osi Y, liczba komórek planszy w osi X, liczba komórek planszy w osi Y.
3. Klient wysyła do serwera informację o początkowym położeniu właśnie zarejestrowanego robota:

- ramka zapytania ma rozmiar 12 bajtów oraz nagłówek oznaczony 6 (`CURRENT_POSITION`), zawiera ona kolejno globalne id robota, położenie robota w osi X (tj. numer komórki, w której robot się znajduje, patrząc względem osi X), położenie robota w osi Y;
- serwer nie odpowiada na to zapytanie.

Zdarzenia 2, 3 powtarzają się tyle razy, ile dostępnych jest robotów, natomiast po zarejestrowaniu pierwszego robota serwer może przejść do wykonywania zdarzenia 4.

4. Serwer wysyła robotowi, wybranemu spośród już zarejestrowanych, polecenie przejechania do jednej z sąsiednich komórek oraz sygnał, czy ma pozwolenie na przejazd do docelowej komórki:
   - ramka zapytania ma rozmiar 28 bajtów oraz nagłówek oznaczony 4 (`RESPONSE_SECTOR`), zawiera ona kolejno globalne id robota, docelowe położenie robota w osi X, docelowe położenie robota w osi Y, flaga oznaczająca pozwolenie na przejazd (1) lub też jego brak (0), numer klienta (w tej chwili ignorowany), docelowe położenie w osi X, docelowe położenie w osi Y;
   - klient nie odpowiada na to zapytanie.
5. a) W przypadku braku pozwolenia na przejazd robot czeka na otrzymanie pozwolenia od serwera, przy czym zadaniem serwera jest monitorowanie faktu, że istnieje robot, który pytał o pozwolenie, dostał odmowę i znajduje się w stanie oczekiwania.

   b) W przypadku otrzymania pozwolenia na przejazd robot przejeżdża do komórki docelowej i w momencie opuszczenia poprzedniej komórki całą swoją powierzchnią informuje serwer o zwolnieniu tejże:
      - ramka zapytania ma rozmiar 16 bajtów oraz nagłówek oznaczony 3 (`REQUEST_SECTOR`), zawiera ona kolejno globalne id robota, poprzednie położenie robota w osi X, poprzednie położenie robota w osi Y, flaga oznaczająca opuszczenie poprzedniej komórki - wartość liczbowa 0;
      - serwer nie odpowiada na to zapytanie.

6 Interfejs graficzny

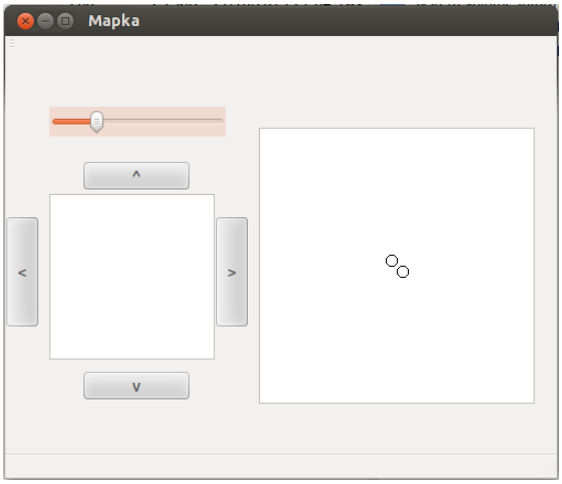
W celu stworzenia GUI wykorzystano biblioteki *Qt*. Interfejs graficzny został zaprojektowany z wykorzystaniem programu *QtDesigner*.

6.1 Wykorzystane klasy

Tworząc interfejs wykorzystano biblioteki:

- **QGraphicsScene** - biblioteka pozwalająca na stworzenie obiektu reprezentującego dwuwymiarową scenę graficzną, na scenie można tworzyć i umieszczać obiekty (pojedyncze piksele, wieloboki), obiekty można poddawać transformacji przy pomocy klasy *QTransform*,
- **QGraphicsView** - biblioteka, przy pomocy której można wizualizować zawartość sceny utworzonej z wykorzystaniem klasy *QGraphicsScene*.

6.2 Proponowany interfejs

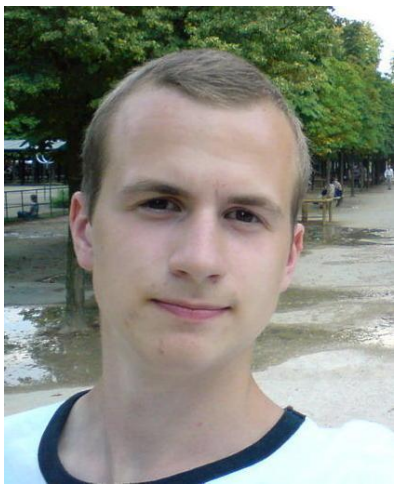


7 Zespół

7.1 Paweł Bogner

Interesuje się elektroniką od strony sprzętowej, szczególnie systemami mikroprocesorowymi. Jego hobby to astronomia.
Wynik testu Belbina: „Racjonalny Analityk”

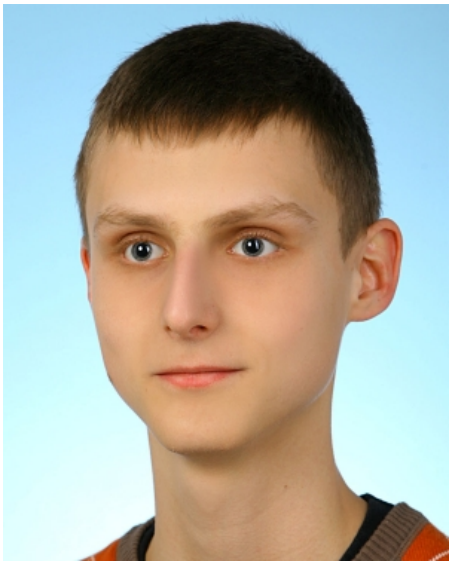
7.2 Marcin Dmochowski



Najbardziej upodobał sobie zadania natury praktycznej i programowanie. Spośród rzeczy niezwiązanych z nauką lubi tenis stołowy i ziemny, brydż, sporty zimowe i wodne,

dobry film i książkę oraz szachy.
Wynik testu Belbina: „Ambitny Komendant”

7.3 Bartosz Folta



Stara się rozwiązać spory drogą dyplomatyczną. Pozytywnie nastawiony do zadań wymagających doszkolenia się w danej dziedzinie.
Wynik testu Belbina: „dusza zespołu”

7.4 Grzegorz Maj

Zainteresowania: góry, w szczególności wspinaczka i skitouring.
Wynik testu Belbina: „Racjonalny Analityk”

7.5 Krzysztof Nomejko

Nieoceniony w każdej grupie. Bardziej praktyk niż teoretyk.
Wynik testu Belbina: „Dusza Zespołu”