AUTOMATYKA • 2009 • Tom 13 • Zeszyt 3

Sławomir Jeżewski*, Maciej Łaski*

Przegląd i porównanie środowisk symulacji robotów mobilnych

1. Wprowadzenie

Symulacja i wspomaganie projektowania odgrywa coraz poważniejszą rolę we współczesnym przemyśle. Środowiska CAD/CAM i środowiska symulacyjne z powodzeniem stosowane są w przemyśle maszynowym, samochodowym i lotniczym zmniejszając koszty opracowania rozwiązań, przyspieszając termin wprowadzenia produktu do użytku. Podobnie jest w zastosowaniach robotycznych, w których zadanie konstrukcji robota zawiera w sobie etapy projektowania mechanicznego, elektrycznego i informatycznego. Etapy te można zreorganizować, przyspieszyć i zrównoleglić, stosując narzędzia CAD/CAM w połączeniu z narzędziami symulacyjnymi. Symulacja robota i jego otoczenia pozwala na znacznie szybsze testowanie algorytmów zachowania robota w różnych sytuacjach, czasem niedostępnych w rzeczywistym otoczeniu. Dzięki symulacji zmniejszane są koszta testowania i sprzętu, oraz zmniejszane jest ryzyko uszkodzenia sprzętu w wyniku nieprawidłowego zachowania robota podczas pierwszych faz uruchomienia. Praca z symulatorem wyklucza podstawowe źródła błędów z algorytmów i dostarcza danych dla usprawnienia konstrukcji mechanicznej i elektronicznej robota.

Literatura naukowa na temat tworzenia symulatorów robotów mobilnych i wymagań stawianych symulatorom jest stosunkowo uboga. Pojedyncze publikacje pochodzą głównie ze środowiska twórców symulatorów [1, 2]. Szersza aktywność publikacyjna związana jest z programami CAD/CAM w zastosowaniach robotycznych [3–5]. W niniejszej publikacji zawarto wyniki przeglądu dostępnych środowisk symulacji robotów mobilnych. Sformułowano oczekiwania w stosunku do środowiska symulacyjnego. Opracowanie ma po części charakter przewodnika dla projektantów oprogramowania i konstruktorów robotów, którzy chcą wykorzystać potencjał dostępnych narzędzi.

2. Kryteria doboru środowiska symulacji

W trakcie projektowania robota, a w szczególności robota mobilnego, występuje wiele stałych i charakterystycznych punktów projektu. Pierwszym etapem jest zaprojektowanie

* Katedra Informatyki Stosowanej, Politechnika Łódzka

konstrukcji mechanicznej robota na podstawie postawionych *a priori* wymagań, zaprojektowanie elektroniki sterującej, a w etapie końcowym oprogramowania. Z oczywistych przyczyn większość prac musi przebiegać etap po etapie. Wdrażanie algorytmów może się odbywać po zaprojektowaniu, wykonaniu i uruchomieniu mikroprocesorowych układów wykonawczych. Systemy mikroprocesorowe mogą być uruchamiane w obecności minimalnego oprzyrządowania mechanicznego.

Kolejność etapów upodabnia projekt konstruowania robota do kaskadowego modelu i powoduje, że większość problemów i niedociągnięć projektowych ujawniać się będzie dopiero na końcowym etapie prac. Część założeń projektowych postawionych *a priori* wykaże swą nierealność lub niemożliwość realizacji dopiero na etapie tworzenia oprogramowania (problem pierwszego typu). Niektóre założenia determinujące początkowe prace konstrukcyjne powstaną dopiero po opracowaniu i przeanalizowaniu prototypów oprogramowania (problem drugiego typu). Stosując oprogramowanie symulacyjne i oprogramowanie CAD/CAM w roli narzędzia prototypującego, można uniknąć problemów obydwu typów.

Przyjmując, że oprogramowanie symulacyjne będzie stanowić ośrodek działań zespołu projektowego i płaszczyznę prototypowania dla algorytmów i konstrukcji mechanicznej robota, należy zadbać, by środowisko symulacji charakteryzowało się określonymi cechami. Według autorów publikacji istotne jest by środowisko:

- Umożliwiało zrównoleglenie prac poszczególnych zespołów projektowych.
- Stanowiło platformę „łączącą" wysiłki wielu grup projektowych.
- Umożliwiało pracę wielodostępną i pracę zdalną, kontrolę wersji.
- Umożliwiało pracę w środowisku rozproszonym tak, by wielkość modelu nie stawiała bariery dla zadań projektowych.

Aby symulacja była skuteczna od strony fizycznej, należy zapewnić:

- możliwość tworzenia/importowania z oprogramowania CAD/CAM trójwymiarowego modelu robota lub dowolnego jego elementu;
- możliwość importowania parametrów masowych i geometrycznych dostępnych jako wyniki projektowania CAD/CAM;
- możliwość symulacji tarcia statycznego, dynamicznego oraz tarcia w układach hydraulicznych;
- możliwość symulacji dynamiki bryły sztywnej, kinematyki i statyki połączeń przegubowych.

Symulowanie zachowania się robota mobilnego w otoczeniu wymaga by możliwe było:

- tworzenie trójwymiarowych modeli otoczenia;
- symulowanie działania oprogramowanego modelu w czasie rzeczywistym;
- symulowanie interakcji robota z otoczeniem;
- symulowanie pracy czujników: ultradźwiękowych, skanerów laserowych 2D i 3D, GPS-u, żyrokompasu, akcelerometrów, kamer.

Oprogramowanie symulacyjne może się stać ośrodkiem prac połączonych zespołów projektowych, jeśli programiści będą mogli zaimplementować dowolnie zdefiniowane bryły/obiekty i dowolne ich zachowania. W tym celu wymaga się:

- możliwości podłączania modułów programowych korzystających ze zmiennych stanu symulatora i mogących wpływać na obiekty elementarne symulatora;
- możliwości instalowania modułów zmieniających charakterystyki obiektów wbudowanych np. moduł implementujący tarcie podłoża o niestandardowej charakterystyce;
- symulacji w środowisku wieloprocesorowym lub wielokomputerowym;
- możliwości wykorzystania akceleratorów sprzętowych np. Nvidia PhysX.

W następnych rozdziałach porównamy znalezione środowiska symulacyjne i zaproponujemy te, które mogły by nadawać się do celów symulacji robota mobilnego lub spełniają choćby niektóre z ww. wymagań.

3. Istniejące środowiska symulacji – krótka charakterystyka

W niniejszym rozdziale przedstawiono listę dostępnych komercyjnie środowisk do symulacji robotów mobilnych. Część z wymienionych niżej narzędzi jest ogólnie dostępna na prawach oprogramowania open source, lub oprogramowania darmowego, część jest składnikami pakietów komercyjnych.

- Aerosim Blockset [6] – Biblioteka dla programu Matlab/Simulink dostarczająca komponenty umożliwiające projektowanie nieliniowych 6-DOF modeli samolotów. Posiada narządza symulujące atmosferę, wiatr, turbulencje i modelujące ziemię (geoida, grawitacja, pole magnetyczne). Istnieje darmowa wersja akademicka.
- ARS MAGNA: Abstract Robot Simulator [7] – Dostarcza abstrakcyjny model świata w którym użytkownik może kontrolować robota mobilnego. Program pisany w języku LISP dla środowisk linuksowych na licencji GPL.
- Biorobotic Simulation Program [8] – Język skryptowy o nazwie Config++ używany do symulacji dynamiki wielosegmentowych robotów. Licencja tylko dla użytkowników niekomercyjnych.
- BugWorks [9] – Dwuwymiarowa symulacja robota. Robot konstruowany jest w wizualnym edytorze. Oferuje możliwość wbudowania w robota czujników zbliżeniowych (odległości, zderzaki, itp.). Dostępny jest także język skryptowy pozwalający na proste obliczenia numeryczne, zawiera instrukcje warunkowe i skoki bezwarunkowe, które mogą realizować pętle. Wersja darmowa pozwala na pełne wykorzystanie możliwości symulacji, ale nie pozwala na zapisywanie i ładowanie wcześniej przygotowanych projektów.
- RapSim [10] – Środowisko do modelowania robotów przemysłowych.
- DYNAMECHS [11] – Silnik fizyki i dynamiki brył sztywnych napisany całkowicie w C++. Pozwala na generowanie trójwymiarowego otoczenia oraz modelowania robotów. Oparty na licencji GNU GPL.

- DynawizXMR [12] – Symulator dynamiki napisany w środowisku Matlab/Simulink. Wersja komercyjna. Wizualizacja w postaci wykresów robionych w Matlabie.
- EASY-ROB [13] – Program służący do modelowania robotów przemysłowych. Produkt komercyjny.
- eyeSim [14] – Symulator robotów mobilnych. Udostępnia fizykę platformy z kołami dyferencyjnymi, szereg czujników i wizualizację trójwymiarową.
- eyeWyre Simulation Studio [15] – Środowisko symulacji i modelowania robotów. Możliwość modelowania dynamiki oraz zachowania czujników. Projekt komercyjny.
- The FLAT 2-D Robot Simulator [16] – Symulator robota RWI[1] w środowisku dwuwymiarowym. Program działa w środowisku Linux. Program komunikujący się z robotem może być napisany w Języku LISP.
- Gwelf [17] – Symulacja robota przemysłowego i jego interakcji z otoczeniem. Licencja GNU GPL.
- Juice [18] – Trójwymiarowe środowisko symulacji dynamiki robota przy użyciu języka VPL[2]. Brak możliwości interakcji z otoczeniem (brak czujników).
- Khepera Simulator [19] – Dwuwymiarowy symulator robota Khrpera[3]. Środowisko dostępne pod systemem Linux. Licencja dla użytku niekomercyjnego.
- Kinlab [20] – Modelowanie dynamiki robota przemysłowego. Oprogramowanie oparte na Java i Java3D. Oprogramowanie darmowe.
- LME Hexapod [21] – Środowisko symulacji robota sześcionożnego. Dołączone są źródła w języku C++. Licencja GNU GPL.
- Microsoft Robotics Developer Studio 2008 [22] – RDS jest rozbudowanym środowiskiem symulacji robotów przeznaczonym dla hobbystów, studentów i użytkowników komercyjnych. Oparty jest na technologii dostarczania usług (w stylu REST). Łączenie się z usługą dostarcza odpowiedniej funkcjonalności, która pozwoli symulować, lub sterować rzeczywistym lub symulowanym robotem. Oprogramowanie dostępne jest nieodpłatnie w wersji akademickiej, przeznaczonej dla studentów oraz wersji Express, która nie posiada pełnej funkcjonalności.
- MOBS [23] – Symulator istniejącego robota „Robuter II". Środowisko pozwala na modelowanie zachowania czujników i widok z kamery robota. Projekt napisany w języku C. Użytkownik ma możliwość tworzenia własnego trójwymiarowego otoczenia.
- OpenDynamicEngine [24] – Niezależna od platformy biblioteka napisana w języku C++ do symulacji dynamiki ciała sztywnego, pojazdów naziemnych, robotów i obiektów ruchomych. Obsługuje dynamikę ruchu stawów, tarcie i detekcje kolizji. Oparte na licencji GNU GPL.

---

[1] RWI – Robot dwukołowy posiadający jeden lub dwa skanery laserowe i dwanaście sonarów.
[2] Visual Programming Language
[3] http://www.k-team.com/kteam/home.php?rub=0&site=1&version=EN

- POPBUGS [25] – Środowisko symulacji dwuwymiarowej przestrzeni przeznaczonego dla robotów posiadających bezwładność. System operacyjny Linux.
- ROBOOP [26] – Obiektowo zorientowana biblioteka do symulacji robotów napisana w języku C++. Implementuje kinematykę prostą i odwrotną, przyspieszenia itp. Licencja GNU GPL.
- RoBOSS [27] – System do symulacji i testowania konstrukcji oraz algorytmów dla robotów. Umożliwia budowanie robotów o dowolnych kształtach i właściwościach. Symuluje różne połączenia pomiędzy częściami robotów: siłowniki, osie, itp oraz elementy aktywne, jak silniki, serwomechanizmy, oraz sensory. System zbudowany z mniejszych modułów, z których każdy może zostać uruchomiony na osobnym komputerze. Dostępne języki programowania C++ i C#. Dostępne są źródła oraz skompilowane wersje programu. Brak wzmianki o typie licencji.
- RobotWorks [28] – Interaktywne środowisko CAD do modelowania i symulacji ruchu robotów przemysłowych. Rozwiązanie komercyjne.
- RoboWorks [29] – Narzędzie do modelowania i symulowania systemów fizycznych. Pozwala na modelowanie dowolnego obiektu (robota). Udostępnia interfejs programistyczny do wykorzystania w językach umożliwiających korzystanie z bibliotek dll. Wersja komercyjna.
- SodaConstructor [30] – Symulator oparty na Javie udostępniany poprzez interfejs WWW. Umożliwia modelowanie robotów i symulację dwuwymiarową. Nie posiada implementacji pełnego symulatora dynamiki i kinematyki.
- Webots [31] – Pozwala na tworzenie trójwymiarowych „światów" zasiedlonych wieloma symulowanymi robotami. Wspiera tworzenie kołowych, nożnych lub latających robotów. Zawiera API do kontroli robotów, pełną fizyką opartą na Silniku ODE (Open Dynamic Engine), bardzo dużą bibliotekę modeli czujników, skanerów, kamer i urządzeń komunikacyjnych. Rozwiązanie komercyjne działające pod systemami Windows, Linuks i MAC OS X.
- Yobotics! Simulation Construction Set [32] – Symulator dynamiki ciała. Przeznaczony do symulacji robotów kroczących: humanoidów, robotów sześcionożnych, rąk bądź chwytaków i innych mechanizmów.

4. Porównanie dostępnych środowisk symulacji robotów

Analiza właściwości wyżej wymienionych środowisk symulacji wykazuje, że nie wszystkie spełniają wyznaczone w rozdziale drugim kryteria. Niektóre nie zawierają zaimplementowanej dynamiki bryły sztywnej lub nie potrafią wizualizować danych trójwymiarowych. W tabelach 1 i 2, zostało przedstawione zestawienie tych środowisk, które spełniają podstawowe kryteria.

Tabela 1
Porównanie wybranych środowisk symulacji robotów

Nazwa środowiska	Wbudowane wsparcie dla grupowego projektowania	Działanie w środowisku rozproszonym	Wsparcie dla istniejących środowisk CAD/CAM	Modelowanie otoczenia i interakcji z nim
DYNAMECHS				TAK
eyeSim				TAK
eyeWyre Simulation Studio				TAK
Microsoft Robotics Developer Studio 2008	TAK	TAK	TAK*	TAK
OpenDynamicEngine				TAK
ROBOOP				TAK
RoBOSS		TAK		TAK
RoboWorks				TAK
Webots				TAK

* Wspierany format obj i collada

Tabela 2
Porównanie wybranych środowisk symulacji robotów (kontynuacja)

Nazwa środowiska	Wizualizacja 3D	Kinematyka stawów	Dynamika bryły sztywnej	Gotowe biblioteki czujników	Sprzętowe wsparcie obliczeń fizycznych	Darmowe bądź studencka wersja
DYNAMECHS	TAK	TAK	TAK			TAK
eyeSim			TAK	TAK		TAK*
eyeWyre Simulation Studio	TAK	TAK	TAK	TAK		
Microsoft Robotics Developer Studio 2008	TAK	TAK	TAK	TAK	TAK	TAK
OpenDynamicEngine	TAK	TAK	TAK			TAK
ROBOOP	TAK	TAK				
RoBOSS	TAK	TAK	TAK	TAK		**
RoboWorks	TAK	TAK				
Webots	TAK	TAK	TAK	TAK		

* Do rozwiązań niekomercyjnych

** Brak wzmianki o typie licencji

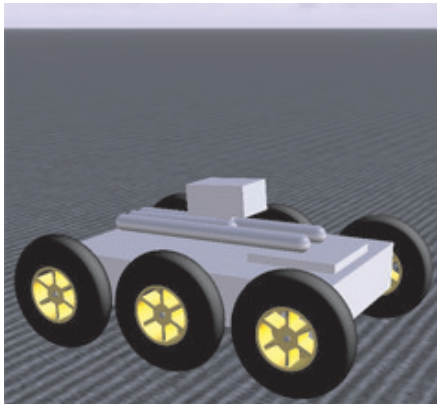
5. Projekt robota mobilnego w środowisku Microsoft Robotics Developer Studio

Spośród wszystkich przedstawionych środowisk symulacji, Microsoft Robotics Developer Studio jest jednym z najbardziej funkcjonalnych i rozbudowanych symulatorów. Posiada wsparcie dla projektowania grupowego i może zostać uruchomione w środowisku rozproszonym, dzięki czemu można przyspieszyć pracę w zespole projektowym. Jako jedyne wspiera sprzętowe obliczenia fizyczne Nvidia PhysX, oraz spełnia podstawowe wymagania postawione środowiskom symulacji. Analiza pozycji ww. tabelach ukazuje, że środowisko MRDS jest najlepszym narzędziem do symulacji robota mobilnego i dlatego zostało wykorzystane przez autorów w projekcie.

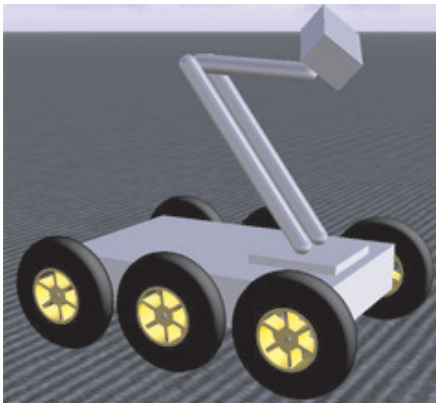
Symulowany robot to sześciokołowa platforma ze sztywnym zawieszeniem umożliwiająca niezależne sterowanie każdym z kół. Dodatkowo do platformy przymocowane jest ramię z dwoma stopniami swobody oraz głowica znajdująca się na końcu ramienia, również z czterema stopniami swobody. Robot wyposażony jest w szereg czujników i kamery, które także są zamodelowane i wprowadzone do systemu symulacyjnego. Wśród czujników znajdują się: skaner laserowy typu SICK, dalmierze ultradźwiękowe, kamery w układzie stereowizyjnym oraz kamer obserwacyjne.

Założeniem symulacji jest przetestowanie robota i jego możliwości w środowisku podobnym do rzeczywistego. Środowisko ma zapewniać symulację podstawowych zjawisk fizycznych, dać możliwość kontroli nad częściami ruchomymi robota (silniki napędzające koła i elementy ramienia pokazano na rys. 1 i 2) oraz pobierać dane z symulowanych czujników. Algorytmy sterujące będą implementowane w języku C lub C++ jako zewnętrzne programy komunikujące się z symulatorem za pomocą dowolnego protokołu. Na rysunku 3 pokazano wysokopoziomową koncepcję środowiska symulacyjnego. Sercem symulacji jest środowisko Microsoft Robotics Developer Studio z modelem robota i otoczenia. Natomiast MODUŁY DECYZYJNE są to wcześniej wspomniane programy lub elektroniczne sterowniki, które będą nadzorowały pracę robota oraz zbierały i przetwarzały dane pochodzące z symulacji. Pozwoli to na późniejsze przeniesienie kodu algorytmów na procesory docelowe znajdujące się w rzeczywistym robocie. Z tego powodu interfejs komunikacyjny symulatora musi być zgodny z interfejsem poszczególnych urządzeń znajdujących się w robocie.

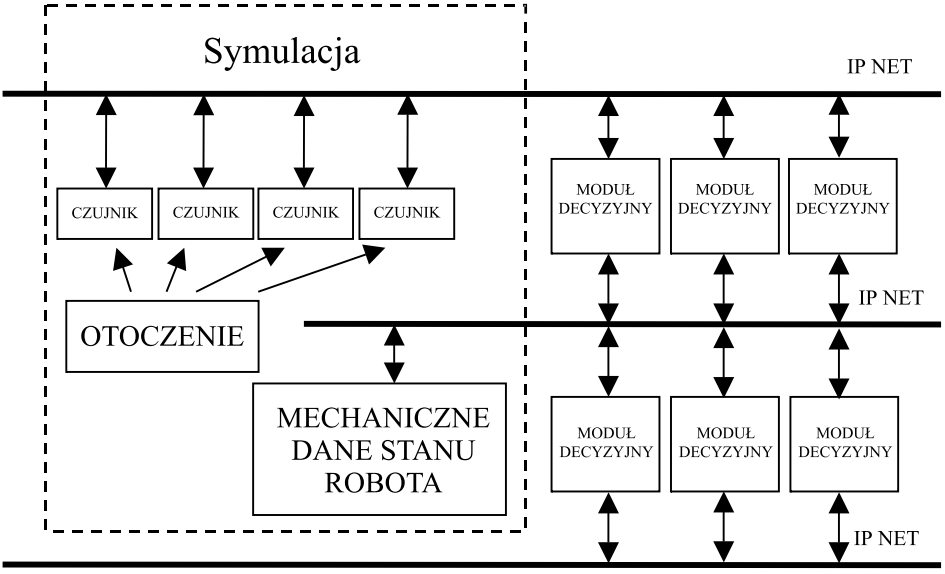
Do zagadnień symulacyjnych zdecydowano się wykorzystać protokół TCP/IP, gdyż jest łatwo dostępny w komputerach typu PC oraz jest zgodny z dyrektywą JAUS (*Joint Architecture for Unmanned Systems*), która została opracowana przez DARPA na potrzeby robotyki wojskowej. Każdy moduł robota otwiera swoje gniazdo komunikacyjne, do którego będą łączyć się zewnętrzne programy sterujące. Modułami, które będą otwierać swoje porty, są: podstawa (sterowanie silnikami kół), ramię (sterowanie silnikami ramienia), kamery, czujniki ultradźwiękowe, skaner laserowy typu SICK. W ten sposób można testować każdy moduł z osobna i dowolnie sterować strumieniem danych.



Rys. 1. Symulowany robot



Rys. 2. Symulowane ramię



Rys. 3. Schemat symulacji

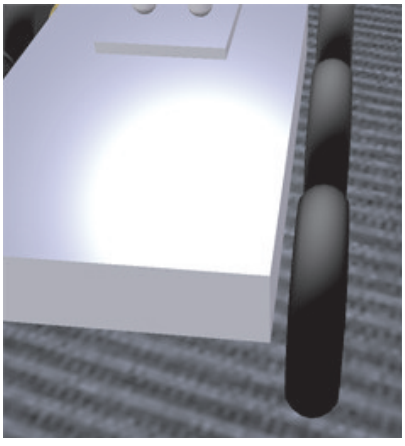
6. Problemy implementacyjne dostrzeżone w środowisku Microsoft Robotics Developer Studio

W procesie tworzenia symulacji rzeczywistego robota w środowisku Microsoft Robotics Developer Studio natrafiliśmy na problemy, które uniemożliwiły przeprowadzenie symulacji w założonych warunkach. Środowisko to jest stosunkowo nowym oprogramowa-

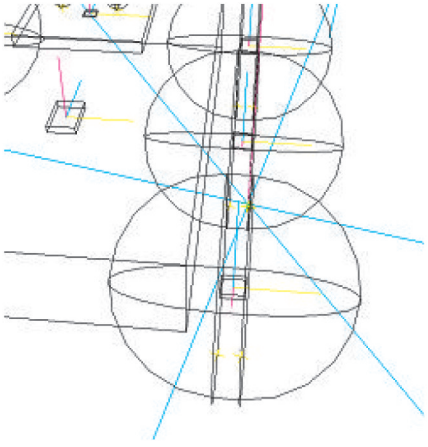
niem i zawiera jeszcze wiele niedociągnięć oraz słabą dokumentację. Poniżej przedstawimy te, na które natrafiliśmy w trakcie procesu implementacji symulatora naszego rzeczywistego robota. Jednym z nich jest symulacja modelu tarcia koła, drugim jest powstawanie dodatkowych sił w silniku fizyki wywoływanych próbą poruszania ramieniem do pozycji kolizyjnych.

6.1. Model tarcia koła

Silnik PhysX dostarcza zaimplementowany model koła (WheelEntity), którego podstawowymi właściwościami są promień, prędkość kątowa oraz kąt skrętu koła (potrzebny do symulacji samochodu). Dla koła dostępny jest bardzo rozbudowany model tarcia. Można symulować tarcie wzdłużne i boczne w zależności od potrzeb. Model ten ma jednak dużą wadę, ponieważ jest nieparametryczny i nie zależy od współczynnika tarcia podłoża. W praktycznych zadaniach projektowych autorzy publikacji założyli, że robot będzie poruszał się po różnorodnym podłożu oraz powinien reagować na zmieniające się charakterystyki podłoża. Dlatego opracowano zamienny model koła oparty na cylindrycznym kształcie, połączony z ciałem robota za pomocą obrotowego połączenia. Dzięki nowemu modelowi można uzyskać różny współczynnik tarcia w zależności od podłoża, na którym robot się porusza. Model tego tarcia zawiera uproszczenia, jednak na potrzeby testowania i symulacji jest wystarczający.



Rys. 4. Symulacja koła



Rys. 5. Widok z silnika fizyki

Przy realizacji modelu koła natrafiono na dodatkowy problem: środowisko symulacyjne MRDS nie ma zaimplementowanej bryły o kształcie cylindra, ale występuje tam kształt kapsuły (walec zakończony z obu stron kulistymi kopułami). Jest to jedyny kształt, który może posłużyć do symulacji koła. Wprowadza on pewne błędy związane z szerokością ro-

bota oraz faktem, że fizyczna reprezentacja koła wtapia się w ciało robota, przez co pojawia się problem z ustawianiem fizycznych modeli czujników i ramienia blisko bocznych krawędzi obudowy.

6.2. Dodatkowe siły występujące w połączeniach elementów dynamicznych

Podczas symulowania pracy ramienia robota dostrzegliśmy błędne zachowanie silnika fizyki. Próbowaliśmy ustawić ramię robota w konfiguracji, która jest nieosiągalna ze względu na kolizję ramienia z obudową robota (rys. 6).

Rys. 6. Niemożliwa do osiągnięcia konfiguracja ramienia

W tej konfiguracji głowica napiera ze stosunkowo dużą siłą na pozostałe elementy robota. Zgodnie z zasadami fizyki robot powinien pozostawać w stanie spoczynku, ponieważ nie działa na niego żadna zewnętrzna siła. Jednak mechanizm symulacyjny powoduje powolne poruszanie się całego robota do przodu, tak jakby działała na niego zewnętrzna siła.

6.3. Podsumowanie

Pomimo kilku dostrzeżonych niedociągnięć Microsoft Robotics Developer Studio plasuje się na pierwszym miejscu pod względem funkcjonalności oraz stabilności pracy. Gotowe komponenty oraz inne rzeczywiste roboty, takie jak Pioneer 3DX czy LEGO NXT

zawarte w środowisku MRDS, dostarczają dużej ilości wiedzy na temat symulacji zarówno całego robota, jak i jego elementów. Zaletą jest także duża społeczność korzystająca z ogólnodostępnego forum. Największym minusem jest uboga dokumentacja, jednak to zrozumiałe, biorąc pod uwagę krótki staż środowiska, które pojawiło się w 2008 roku.

Podsumowując cechy środowiska Microsoft Robotics Developer Studio, dochodzimy do wniosku, że jest ono odpowiednim narzędziem do przeprowadzenia symulacji robota mobilnego. Dostrzeżone w trakcie pracy środowiska błędy udało się ominąć i opracować pakiet oprogramowania, który pozwoli prototypować procedury, schematy zachowania robota, przeprowadzić wiele testów jeszcze przed powstaniem właściwego robota wraz ze wszystkimi układami elektronicznymi.

Literatura

[1] Koestler A., Bräunl T., *Mobile Robot Simulation with Realistic Error Models*. [w:] International Conference on Autonomous Robots and Agents, ICARA 2004, Dec. 2004.

[2] Brian P., Gerkey R., Vaughan T., Howard A., *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*. [w:] Proceedings of the International Conference on Advanced Robotics (ICAR 2003), 2003.

[3] Pollack J., Lipson H., Funes P., Ficici S., Hornby G., *Coevolutionary Robotics.* IEEE Computer Society, Washington 1991.

[4] Ribeiro F., Norrish J., *Rapid prototyping using robot welding – process description.*

[5] Čuboňová N., Kumičáková D., *Tools and applications of computer aided manufacturing and robot virtual model designing.* [w:] 6th International Multidisciplinary Conference, 2005.

[6] Aerosim_Blockset, *http://www.u-dynamics.com/aerosim/*, luty 2009.

[7] ARS_MAGNA:_Abstract_Robot_Simulator, *http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/testbeds/arsmagna/0.html*, luty 2009.

[8] Biorobotic_Simulation_Program, *http://www.life.uiuc.edu/delcomyn/RSSimSoftware.htm*, luty 2009*l.*

[9] BugWorks, *http://www.cogs.sussex.ac.uk/users/christ/bugworks/*, luty 2009.

[10] RapSim, *http://www.camelot.dk/*, luty 2009.

[11] DYNAMECHS, *http://sourceforge.net/projects/dynamechs/*, luty 2009.

[12] DynawizXMR, *http://www.concurrent-dynamics.com/xmr/*, luty 2009.

[13] EASY-ROB, *http://www.easy-rob.com/en/product.html*, luty 2009.

[14] eyeSim, *http://robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html*, luty 2009.

[15] eyeWyre_Simulation_Studio, *http://www.eyewyre.com/*, luty 2009.

[16] The_FLAT_2-D_Robot_Simulator, *http://www.cs.utexas.edu/users/qr/robotics/flat/*, luty 2009.

[17] Gwell, *http://diablo.ict.pwr.wroc.pl/~pjakwert/*, luty 2009.

[18] Juice, *http://www.natew.com/juice/*, luty 2009.

[19] Khepera_Simulator, *http://diwww.epfl.ch/lami/team/michel/khep-sim/index.html*, luty 2009.

[20] Kinlab, *http://kinlab.dyndns.org/*, luty 2009.

[21] LME_Hexapod, *http://www.laboratoryformicroenterprise.org/lme/ComputerCraftsmanship.html*, luty 2009.

[22] Microsoft_Robotics_Developer_Studio_2008, *http://msdn.microsoft.com/en-us/robotics/default.aspx*, luty 2009.

[23] MOBS, *http://robotics.ee.uwa.edu.au/mobs/*, luty 2009.

[24] OpenDynamicEngine, *http://ode.org/*, luty 2009.

[25] POPBUGS, *http://www.cogs.sussex.ac.uk/users/christ/popbugs/intro.html*, luty 2009.
[26] ROBOOP, *http://www.cours.polymtl.ca/roboop/*, luty 2009.
[27] RoBOSS, *http://home.icslab.agh.edu.pl/~soofka/RoBOSS2/*, luty 2009.
[28] RobotWorks, *http://www.robotworks-eu.com/*, luty 2009.
[29] RoboWorks, *http://www.newtonium.com/*, luty 2009.
[30] SodaConstructor, *http://sodaplay.com/creators/chris_gjersing/items/recycler*, luty 2009.
[31] Webots, *http://www.cyberbotics.com/products/webots/*, luty 2009.
[32] Yobotics!_Simulation_Construction_Set, *http://yobotics.com/simulation/simulation.htm*, luty 2009.