mgr inż. Janusz Będkowski
mgr inż. Grzegorz Kowalski
prof. dr hab. Andrzej Masłowski
Przemysłowy Instytut Automatyki i Pomiarów

ELEMENTY GRAFIKI KOMPUTEROWEJ W PROJEKTOWANIUZAAWANSOWANYCH SYMULATORÓW ROBOTÓW MOBILNYCH

W pracy przedstawiono zastosowanie grafiki komputerowej w projektowaniu zaawansowanych symulatorów robotów mobilnych. Nowoczesne karty graficzne umożliwiają symulację i wizualizację w czasie rzeczywistym złożonych robotów mobilnych. Zastosowanie grafiki komputerowej w tworzeniu wirtualnej rzeczywistości dla modelu robota umożliwia obserwację działania zaawansowanych algorytmów autonomicznych robotów mobilnych. Efekty specjalne w postaci systemu cząstek, odbić, cieni sprawiają, że wirtualna scena wiernie odzwierciedla rzeczywistość. Zastosowanie zaawansowanych symulatorów robotów mobilnych zmniejsza koszt związany z projektowaniem konfiguracji percepcji robota, algorytmów autonomicznego wykonywania zadań, trenowaniem potencjalnych użytkowników.

THE COMPUTER GRAPHIC APPROACH OF THE ADVANCED MOBILE ROBOT SIMULATION DESIGN.

The following paper presents the computer graphic approach of the advanced mobile robot simulation design. Modern graphic computing provides a possibility in approaching the simulation and visualization of the sophisticated mobile robots in advance stage of the design, as well in it's real time mode. The virtual reality which facilitated and composed by the graphic technology allows us to observe the execution of the advanced autonomous algorithms of the mobile robots. The special effects which inherent in the technology such as particle system, reflection, shadows are able to approach the sophisticated scenario designs. This approach systematically is able to provide an effective and efficient production. Therefore, The implementation of the advanced mobile robots simulation decreases the cost of the configuration of the robot perception design, also the cost of the developing the autonomous navigation algorithms and the training customers.

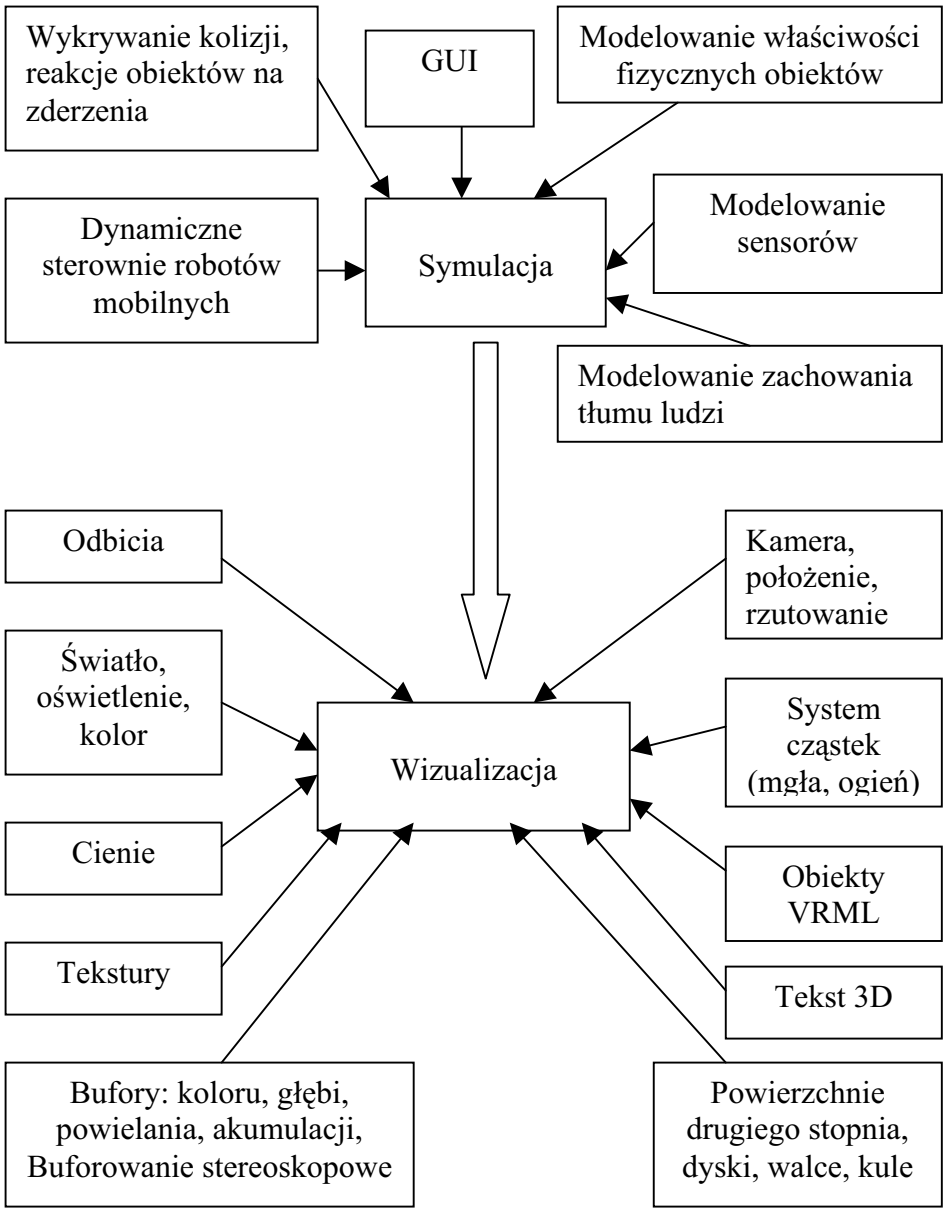
1. WPROWADZENIE

Oprogramowanie symulacyjne jest doskonałym narzędziem wspomagającym użytkowanie robotów mobilnych. Z punktu widzenia pedagogiki symulacja komputerowa jest udowodnioną techniką trenowania [14]. Projektowanie wstępne symulatora robota mobilnego polega na wybraniu odpowiednich technologii potrzebnych do realizacji zadania stawianego przed tworzonym oprogramowaniem. W dobie istnienia wielu dostępnych silników symulatorów i silników fizyki [1, 2, 3] nie trzeba już implementować oprogramowania rozpoczynając od pustej kartki. Doskonałym przykładem jest platforma Player/stage/gazebo [10] oferująca duże możliwości symulacyjne, oraz dostępny silnik fizyki ODE – Open Dynamic Engine [4], umożliwiający implementację algorytmów realizujących zadanie symulowania fizyki obiektów w wirtualnej scenie.

Zastosowanie grafiki komputerowej w projektowaniu zaawansowanych symulatorów robotów mobilnych umożliwia wierne odwzorowanie rzeczywistości. Zastosowanie silnika fizyki umożliwia interakcję obiektów w scenie. Zaawansowany symulator charakteryzuje się wysoką złożonością sceny oraz możliwością interakcji z jej obiektami a także wysoką jakością zastosowanych modeli fizycznych.

2. SYMULATOR

Symulator robota mobilnego składa się z trzech głównych elementów: symulacji modelowanych zjawisk, wizualizacji w czasie rzeczywistym oraz warstwy komunikacji. Rys. 1 przedstawia schemat blokowy symulatora.



Rys. 1. Schemat blokowy symulatora

2.1. Symulacja

Blok symulacji jest odpowiedzialny za obliczenia pozycji robotów, stanu czujników, oraz wykrycia kolizji. Wszystkie operacje powinny być wykonywane w czasie rzeczywistym.

Dodatkowo blok ten aktualizuje pozycje i konfiguracje modeli ludzi w scenie. Podstawowym elementem dynamicznej sceny jest obiekt. Pozycja obiektu zdefiniowana jest jako wektor:

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{1}$$

Orientacja obiektu jest reprezentowana jako macierz 3 x 3 zwana macierzą rotacji R. Macierz rotacji R ma własność, że przekształca punkt a' w lokalnym układzie współrzędnych do punktu a zgodnie z zależnością:

$$a = Ra' + p \tag{2}$$

Macierz R jest macierzą ortonormalną, więc:

$$R^{-1} = R^T \tag{3}$$

Dla danego punktu $u_x = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ wynik mnożenia Ru_x realizuje zadanie przekształcenia współrzędnych punktu u_x we współrzędne globalnego układu współrzędnych.

$$Ru_x = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix} \tag{4}$$

Pierwsza kolumna macierzy R jest przetransformowanym wektorem kierunkowym osi OX, druga kolumna i trzecia odpowiednio OY i OZ. Wynika z tego, że macierz R składa się z trzech wektorów kierunkowych u_1, u_2, u_3, więc:

$$R = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} \tag{5}$$

Liniowa prędkość v punktu p 3 x 1 jest zdefiniowana jako:

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} dp_x / dt \\ dp_y / dt \\ dp_z / dt \end{bmatrix}, \text{ gdzie t oznacza czas.} \tag{6}$$

Prędkość obrotowa obiektu jest definiowana jako wektor

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{7}$$

2.1.1. Wykrywanie kolizji

Wzór płaszczyzny jest określony jako:

$$Ax + By + Cz - D = 0; \tag{8}$$

Trójka liczb <A, B, C> reprezentuje wektor normalny płaszczyzny N. Liczba D reprezentuje odległość płaszczyzny od początku układu współrzędnych.

Jeśli punkt <x, y, z> spełnia równanie płaszczyzny to znaczy, że leży na płaszczyźnie, a w konsekwencji jego odległość od płaszczyzny wynosi 0. Procedura obliczająca odległość punktu od płaszczyzny jest przedstawiona poniżej:

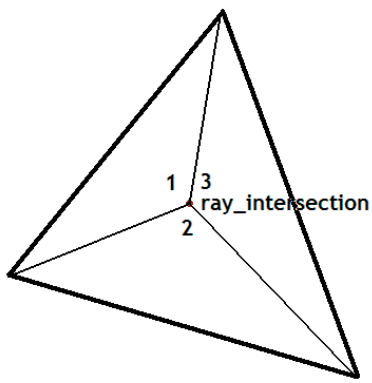
```
procedure DistanceToPlane(point)
{
return    dotProduct(N, point) + D
}
```

Procedura obliczająca punkt przecięcia promienia określonego przez punkt początkowy (ray_position) i kierunek (ray_direction) jest przedstawiona poniżej:

```
procedure RayIntersection(ray_position, ray_direction)
{
     A = dotProduct(N, ray_direction);
     If (A == 0)
          return rey_position;

     return ray_position – ray_direction *
(DistanceToPlane(ray_position) / A)
}
```

Szybka metoda obliczenia czy punkt leży wewnątrz trójkąta polega na sumowaniu kątów 1, 2 i 3 (rys. 2). Jeśli suma tych kątów wynosi 2Π warunek jest spełniony i w konsekwencji wnioskujemy, że punkt leży wewnątrz trójkąta.

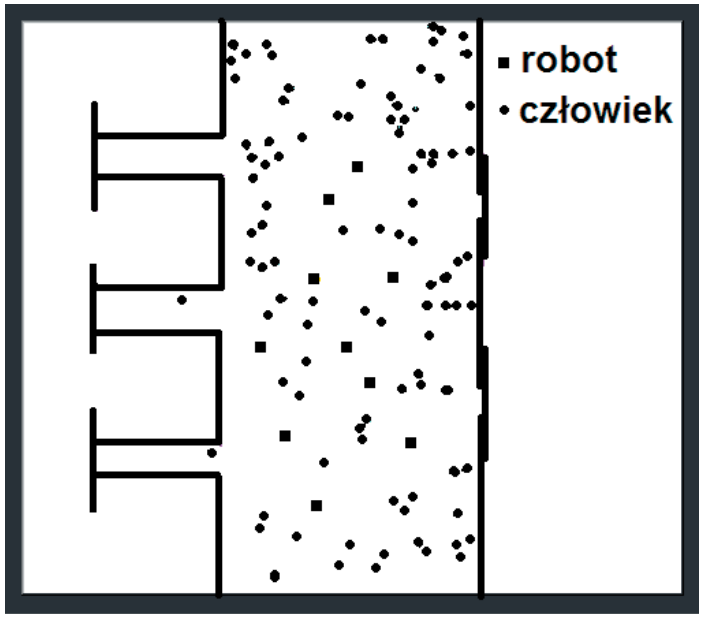


Rys. 2. Punkt leżący wewnątrz trójkąta – punkt przecięcia promienia i płaszczyzny

2.1.2. Modelowanie zachowania ludzi

Zaimplementowano program symulujący spacer przechodniów po budynku. Program ten dokonuje niezbędnych obliczeń związanych wykrywaniem kolizji i postępowego lub obrotowego ruchu człowieka. W kwancie czasu jest generowana kompozycja macierzy tak, aby w czasie rzeczywistym animować ruch. Na ruch człowieka składa się ruch rąk, nóg, ruch postępowy do przodu ruch obrotowy wokoło własnej osi. Program generuje zmianę pozycji do 100 osób, przy czym każda osoba ma inną prędkość zmian pozycji i animacji. Podczas

jednej klatki obliczana jest nowa pozycja człowieka jak i jego kompozycja ciała. W związku z faktem, że program generuje około 30 klatek na sekundę uzyskuje się płynną animację całego zbioru ludzi. Rys. 3 pokazuje działanie programu. Widoczny jest *layout* budynku i pozycja ludzi oraz robotów. Obserwator może zdefiniować obszary tłoku jak i ocenić działanie grupy robotów.



Rys. 3. *Layout* budynku wraz z wizualizacją położenia ludzi i robotów

2.2. Wizualizacja

W ramach bloku wizualizacji wchodzą komponenty odpowiedzialne za generowanie efektów wizualnych. Zastosowanie modelu światła i cieniowania gradientowego uzyskuje się płynne przejście kolorów z jasnego w obszarach oświetlonych do ciemnego w obszarach zaciemnionych. Użycie cieni powoduje zwiększenie wiarygodności wizualizacji. Blok wizualizacji generuje obraz bieżącej sceny i wzbogaca go o efekty specjalne w celu osiągnięcia jak największej wiarygodności imitowanej rzeczywistości.

2.2.1. VRML Virtual Reality Modeling Language

Obiekty w symulatorze projektowane są za pomocą narzędzi umożliwiających eksportowanie ich w formacie VRML. Standard VRML umożliwia opis dynamicznych scen 3D[15], gdzie użytkownik może nawigować po scenie korzystając z przeglądarki internetowej. Sceny VRML mogą być udostępnione w sieci Internet. Dużym plusem zastosowania opisu sceny w postaci pliku VRML jest jego czytelność. Użytkownik może przeczytać zawartość sceny 3D w pliku korzystając z programu do odczytu plików tekstowych.

2.2.2. System cząstek

Przez system cząstek należy rozumieć zbiór pojedynczych elementów zwanych cząstkami [16]. Każda z cząstek może posiadać wartości atrybutów odróżniające ją od innych cząstek. Do atrybutów tych należą: kolor, prędkość, czas życia. Każda cząstka działa autonomicznie, niezależnie od innych cząstek. Cząstki danego systemu posiadają zbiór wspólnych atrybutów. Dzięki temu tworzą wspólnie jeden efekt. Najbardziej powszechnymi zastosowaniami systemu cząstek jest snop iskier, z których każda reprezentowana jest przez jedną cząstkę.

Łącząc zastosowanie systemu cząstek z możliwościami tekstur można tworzyć realistyczne efekty przedstawiające ogień, dym, eksplozję, przepływ cieczy, opady atmosferyczne oraz wszelakie opary.

2.2.3. Odbicia

Za pomocą efektu specjalnego „odbicie" generuje się wirtualny obraz odwrócony danego obiektu. Obraz ten jest wiernym odbiciem obiektu imitującym działanie lustra. Efekt ten występuje często w pomieszczeniach z gładkimi i błyszczącymi powierzchniami.

2.2.4. Bufory koloru, głębi, buforowanie stereoskopowe

W grafice komputerowej podstawowym elementem są bufory specjalnego przeznaczenia. Bufor koloru jest odpowiedzialny za przechowywanie informacji na temat koloru pikseli wyświetlanej sceny. Stosując filtr rozmywający kolory pikseli uzyskuje się efekt nieostrego obrazu. Kolejnym buforem jest bufor głębi który przechowuje informacje o odległości obiektu względem kamery. Bufor ten umożliwia rysowanie obiektów w przestrzeni 3D.

Buforowanie stereoskopowe jest bardzo użyteczną techniką do zaprojektowania symulatora współpracującego z urządzeniami generującymi obraz 3D z rzeczywistą głębią. Przykładem są monitory 3D, a także okulary stereoskopowe. Zastosowanie okularów stereoskopowych jest widoczne w zaawansowanej teleoperacji robotów mobilnych. Symulowanie obrazu stereoskopowego zmniejsza koszty związane z zakupem drogich kamer stereo.

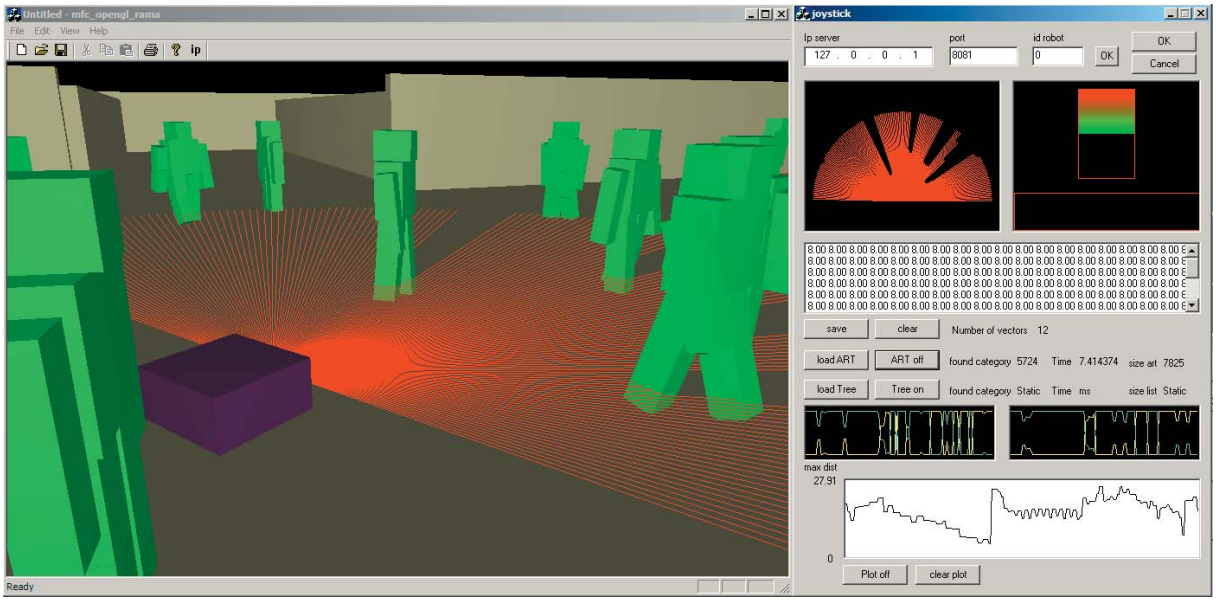
2.3. Warstwa komunikacji

Warstwa komunikacji jest odpowiedzialna za wymianę informacji pomiędzy poszczególnymi komponentami systemu. Zastosowanie technologii CORBA umożliwia programiście wykorzystanie technologii obiektowej w spójnej postaci. Interfejsy obiektów niezależne od sieci i systemów operacyjnych są zdefiniowane przez standard CORBA, wyeliminowana została przez to konieczność integrowania obiektów z interfejsami proceduralnymi.

W ramach przedstawionej pracy połączono wszystkie komponenty systemu za pomocą CORBA TAO, przez co otrzymano jednolitą i spójną architekturę interfejsu sieciowego. Dodanie kolejnej funkcjonalności (np. kolejny czujnik) sprowadza się do zaprojektowania serwera CORBA udostępniającego funkcjonalność obsługiwanego urządzenia. Dzięki tak zorganizowanemu podejściu uzyskuje się spójną architekturę systemu. Podejście komponentowe umożliwia podział symulatora na podzespoły funkcjonalne, co determinuje łatwą rozbudowę systemu. Jednolita architektura programowa umożliwia sprawną pracę nad rozwojem algorytmów sterowania jak i inteligencji obliczeniowej.

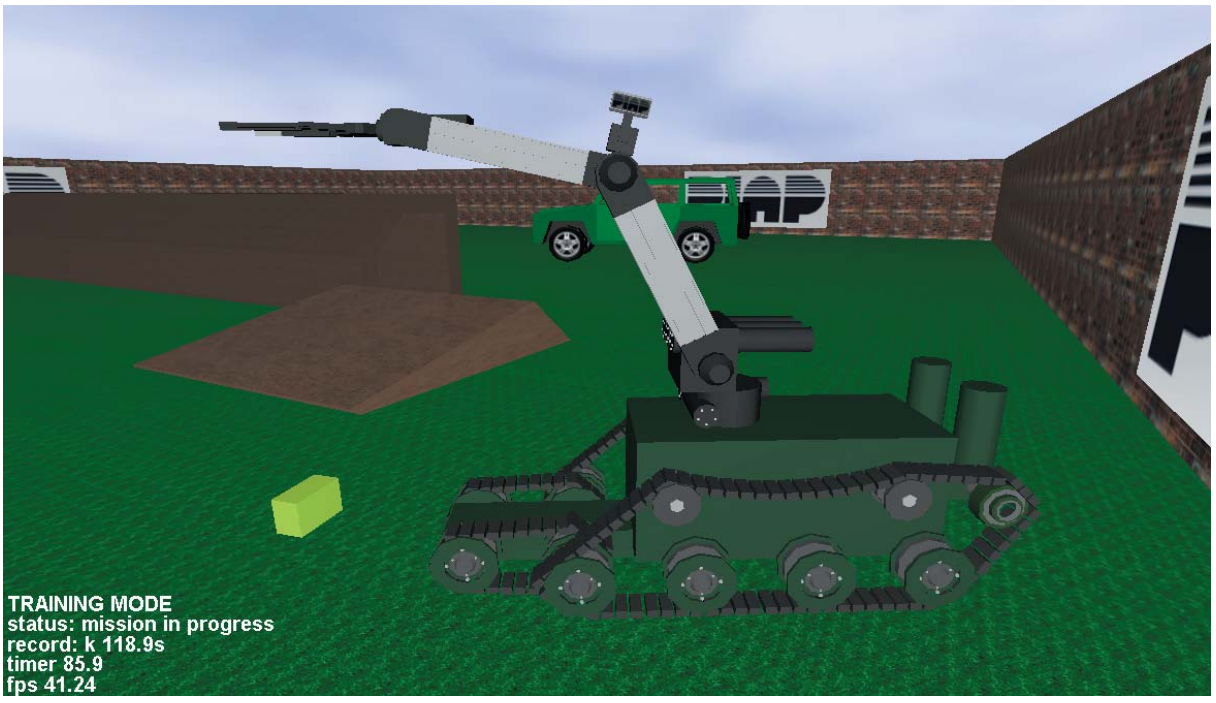
2.4. Przykadowe realizacje

W celu przeprowadzenia eksperymentu związanego z badaniem szybkiego klasyfikatora dla robota mobilnego [17] działającego w obszarach zatłoczonych zastosowano autorski symulator - Multirobot Simulator. Implementacja umożliwia symulacje do 100 spacerujących osób, 10 robotów mobilnych z laserowym systemem pomiarowym. Symulator importuje mapę otoczenia z pliku VRML, dzięki czemu jest możliwe zaprojektowanie dowolnej konfiguracji sceny. Aplikacja działa w czasie rzeczywistym, dzięki zastosowaniu architektury CORBA istnieje możliwość rozproszenia komponentów na różne maszyny PC.



Rys. 4. Multirobot Simulator zaprojektowany do eksperymentów w zatłoczonym środowisku

Rys. 5 przedstawia zrzut z ekranu z dema symulatora robotów INSPECTOR i EXPERT. Oprogramowania umożliwia symulację teleoperacji robotów. Dostępne są widoki ze wszystkich kamer robota. Symulowane są zakłócenia towarzyszące z bezprzewodowym przesyłaniem analogowego sygnału wizyjnego. Istnieje możliwość interakcji robota ze sceną, robot może podnieść bombę a także wybić szybę w samochodzie. Program może być wykorzystany w szkoleniu personelu.



Rys. 5. Zrzut z ekranu dema symulatora robotów INSPECTOR i EXPERT

3. PODSUMOWANIE

Szkolenie personelu za pomocą narzędzi symulacyjnych jest uzasadnione dydaktycznie. Na rynku oprogramowania istnieje wiele komercyjnych i darmowych komponentów symulatorów. W związku z tym, projektowanie zaawansowanych symulatorów robotów mobilnych nie jest już czasochłonne. Wiedza z podstawowych technik grafiki komputerowej pozwala na przygotowanie wiernie odzwierciedlającej rzeczywistość sceny. Wszystkie dostępne silniki gier czy symulacji udostępniają mechanizmy korzystające podstawowych elementów grafiki komputerowej. Istnieje możliwość zaprojektowania sceny z ogniem, dymem, lustrami (błyszczące powierzchnie). Dodatkowo za pomocą bufora kolorów i operacji konwolucji z jądrem filtru rozmywającego można symulować działanie kamer video. Zastosowanie grafiki interaktywnej udostępnia możliwość budowy wirtualnych pulpitów sterowniczych aktywowanych za pomocą myszy komputerowej. Wszystkie wyżej wymienione elementy powodują zmniejszenie kosztów trenowania personelu i przyśpieszają proces szkolenia.

Praca naukowa finansowana ze środków na naukę w latach 2007-2010 jako projekt rozwojowy numer 0046/R/T00/2007/04

4. REFERENCJE

[1] P. Prasithsangaree, J. Manojlovich, S. Hughes, and M. Lewis, *Utsaf: A multi-agent-based software bridge for interoperability between distributed military and commercial gaming simulation*, Simulation, vol. 80, no. 12, pp. 647–657, 2004.

[2] Microsoft, "Microsoft flightsimulator." [Online]. Available: http://www.microsoft.com/games/flightsimulator/

[3] "Webots." [Online]. Available: http://www.cyberbotics.com/products/webots/

[4] "Open dynamics engine." [Online]. Available: http://www.ode.org/

[5] "Simbad 3d robot simulator." [Online]. Available: http://simbad.sourceforge.net/

[6] "eyewyre studio." [Online]. Available: http://www.eyewyre.com/

[7] "Microsoft robotics studio." [Online]. Available: http://msdn.microsoft.com/robotics/

[8] "Matlab." [Online]. Available: http://www.mathworks.com

[9] "Unity." [Online]. Available: http://www.unity3d.com

[10] "Player/stage/gazebo." [Online]. Available: http://sourceforge.net/projects/playerstage

[11] "Simrobot." [Online]. Available: http://www.informatik.uni-bremen.de/simrobot/

[12] "Openscenegraph." [Online]. Available: http://www.openscenegraph.org/

[13] "Newton game dynamics engine." [Online]. Available: http://www.newtondynamics.com/

[14] M. Prensky, *Digital game-based learning*, Book Synopsis, Games2train, pp. 21–21, October 2003.

[15] Tangyun Dai, Zemin Wang, Shouheng Xu, *Research of Creating and Fetching 3D Models of Virtual Reality Based on OpenGL,* Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation June 25 – 26, 2006, Luoyang, China

[16] K. Hawkins, D. Astle *OPENGL programowanie gier,* helion 1999.

[17] J. Będkowski, S. Jankowski, *Multirobot symulator ze sterownikiem robotów opartym na szybkim hybrydowym klasyfikatorze DTFAM*, Przegląd Techniczny 22 2007