Marcin Apostoluk
Wydział Informatyki I Zarządzania
Politechnika Wrocławska

Wrocław, 15.01.2006

Boty w grach komputerowych

Abstract

Gry komputerowe odgrywają znaczącą rolę w dziedzinie sztucznej inteligencji. Oferują świetne środowisko testowe nowych pomysłów i metod. Idealnie nadają się do weryfikacji nowych idei. Tematem bardzo popularnym w ostatnich latach są badania i prace prowadzone nad inteligencją przeciwników. Niniejsza praca przedstawia zagadnienie botów w grach komputerowych 3D skupiając się na metodzie ich tworzenia oraz sposobie funkcjonowania.

1. Podejścia w tworzeniu botów

Rozważając stworzenie inteligentnego przeciwnika(bota) do gry komputerowej należy w pierwszym kroku określić sposób, w jaki bot będzie włączany do gry oraz sposób, w jaki będzie się komunikował z grą.

Istnieją dwa odmienne podejścia do dołączania botów do gry komputerowej[3]:

a) **Programowanie po stronie serwera.**
Bot posiada bezpośredni dostęp do parametrów gry. Najczęściej jest to pełny dostęp do wszystkich informacji(mapa, rozmieszczenie elementów itp.). Dużym problemem jest to, że błąd w programie bota powoduje błąd w całej aplikacji. Ponadto, jeśli tworzy się bota do istniejącej już gry, czasem trudno jest zorientować się w sporym kodzie dostarczanym przez producenta

b) **Programowanie po stronie klienta.**
Bot nie posiada bezpośredniego dostępu do parametrów gry, dysponuje jedynie informacjami przesłanymi przez serwer(zazwyczaj niepełnymi). Plusem jest fakt, że błąd w programie bota nie powoduje błędu w całej aplikacji, dodatkowo boty do istniejącej już gry tworzy się znacznie łatwiej (mniejsza ilość kodu do opanowania).

2. Wymagania stawiane botom

Od bota zawsze wymaga się niskiego zużycia procesora oraz pamięci. Zazwyczaj logika działań bota nie może pochłaniać więcej niż 10-15% czasu

procesora. Jest to ograniczenie bardzo restrykcyjne gdyż w grze zazwyczaj występuje wiele botów jednocześnie. Zużycie procesora powinno być również tak stałe, jak to tylko możliwe, aby nie powodować przestojów w grze komputerowej. Rozważając wymagania pamięciowe, zakłada się, że wszystkie boty jednocześnie nie mogą zużywać więcej niż kilka-kilkanaście megabajtów pamięci.

Wymagania techniczne nie są jedynymi wymaganiami, jakie spełniać muszą boty. Znacznie ważniejsze jest, by bot jak najdokładniej imitował zachowanie gracza ludzkiego. Bot idealny to taki, którego nie można rozróżnić od innego gracza komputerowego. Tak więc bot musi poruszać się po planszy w sposób, w jaki czynią to ludzie, zbierać oraz korzystać ze znalezionych przedmiotów, a także staczać pojedynki z innymi graczami.

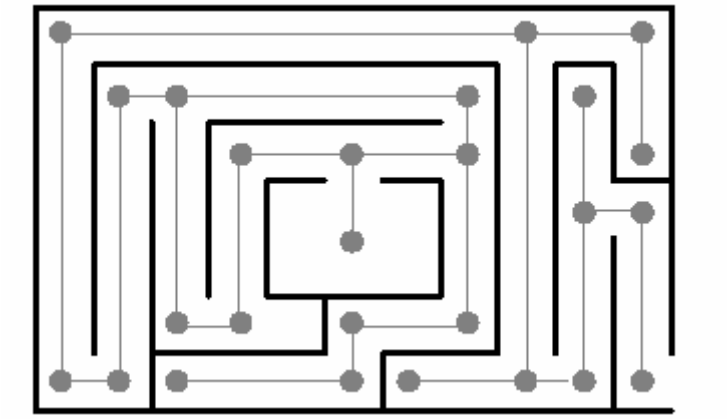
3. Techniki sztucznej inteligencji wykorzystywane w botach

Przy tworzeniu botów stosuje się bardzo wiele technik sztucznej inteligencji, poniżej wymieniono i pokrótce opisano najważniejsze z nich.

Odnajdywanie ścieżek

Bot powinien poruszać się po mapie tak jak człowiek. Istnieje wiele różnych rozwiązań tego problemu. Najczęściej potrzebna jest specjalna reprezentacja mapy. Zazwyczaj stosuje się system punktów kontrolnych (*ang. waypoint*): zbiór punktów i połączeń (często jednokierunkowych) pomiędzy nimi.

System punktów kontrolnych może być tworzony na bieżąco w czasie działania gry, lecz w ten sposób bot prawie nigdy nie dotrze do punktów trudno dostępnych i nie nauczy się efektywnego poruszania po całej mapie. Z tego powodu reprezentacja mapy tworzona jest często tuż przed umieszczeniem bota w grze, nie zaś podczas gry. Manualne tworzenie systemów kontrolnych zazwyczaj odrzuca się z tego powodu, że gracze bardzo często mają możliwość tworzenia własnych map – tak więc rozwiązanie musi być na tyle uniwersalne, że automatycznie dostosuje się do każdej mapy.

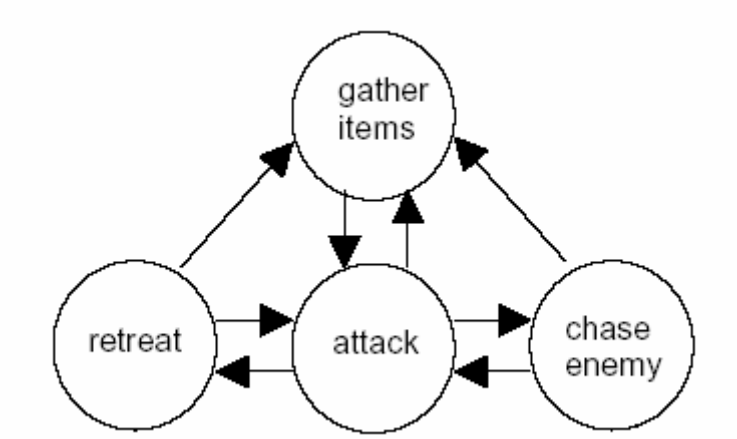


Rysunek 1. Struktura punktów kontrolnych
źródło: [1]

Najczęściej używane algorytmy do odnajdywania ścieżek to: Floyd's, Dijkstra's, A*. Działają one bardzo dobrze na grafach, tak więc idealnie nadają się dla reprezentacji punktów kontrolnych. Ważnym elementem jest także, by algorytm odnalazł kilka ścieżek i aby mógł wybrać jedną z nich (przykładowo nie najkrótszą, lecz najbezpieczniejszą lub przynoszącą największe korzyści).

Automat skończony(*ang. Finite State Machine – FSM*)

Automat z ograniczoną liczbą stanów – automat skończony, to abstrakcyjna maszyna o skończonej liczbie stanów, która zaczyna działanie w pewnym stanie początkowym i zmienia stany przy zajściu pewnych warunków. Za pomocą automatu skończonego łatwo można zamodelować podstawowe czynności wykonywane przez bota. Automat skończony posiada jednak ograniczone możliwości i tylko do pewnej dokładności może odwzorować zachowanie człowieka.



Rysunek 2. Przykładowy automat skończony stanów bota
Źródło: [1]

Logika rozmyta(*ang. fuzzy logic*)

Nadzbiór logiki klasycznej, może operować na pojęciach typu „bardzo wysoki", lub „średnio wysoki". Logika rozmyta jest wykorzystywana w zachowaniu botów do odzwierciedlenia, jak bardzo bot chce wykonywać daną czynność lub posiadać wybrany przedmiot – bazuje np. na wiedzy, w jakim stanie zdrowia jest bot i czy potrzebuje nowej broni. Obliczona zostaje wartość rozmyta(przykładowo: bot bardzo potrzebuje nowej broni).

Sieci neuronowe

Sieci neuronowe używane są w różnych aspektach działania bota, takich jak: zbieranie wiedzy dziedzinowej, lub podobnie jak logika rozmyta – ocenianie jak bardzo bot czegoś potrzebuje. Sieć neuronowa jest zazwyczaj szkolona przed uruchomieniem programu, rzadko w trakcie działania gry (ze względu na czasochłonność obliczeń).

Systemy ekspertowe

Są to systemy przechowujące zdobytą wiedzę uzyskaną z treningu i doświadczenia. Mogą być implementowane do wzbogacenia wnioskowania

bota. Wiedza graczy jest przekazywana do systemu i przechowywana w bazie wiedzy.

Bot może np. używać reguł:
IF bot walczy AND bot ma mało energii THEN bot wycofuje się z walki

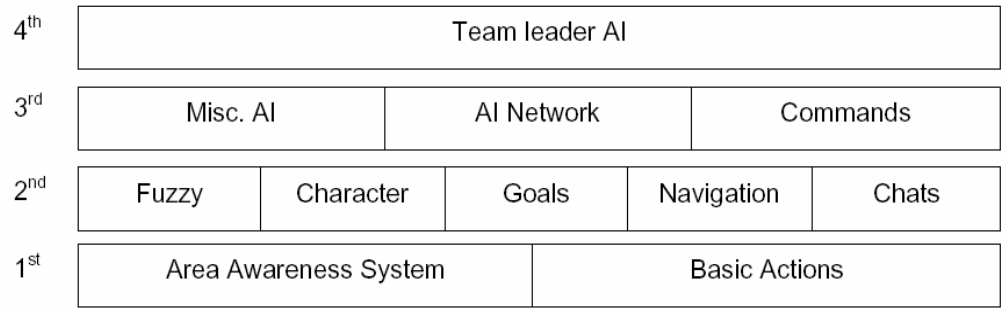
Algorytmy genetyczne

Używane by z populacji botów wybierać i udoskonalać boty najlepsze pod pewnymi względami.

4. Architektura bota

W dalszej części pracy przedstawiona zostanie przykładowa architektura bota pochodzącego z gry komputerowej Quake III®. Informacje na temat tej architektury zostały zaczerpnięte z pracy [1].

Architektura bota z gry Quake III jest 4-warstwowa:



Rysunek 3. Architektura bota
źródło: [1]

Czym wyższa warstwa tym wyższy poziom świadomości bota. Decyzje z warstw wyższych są wykonywane przez warstwy niższe. Warstwy niższe dostarczają danych niezbędnych do podejmowania decyzji warstwom wyższym.

Poniżej przedstawiono krótki opis powyższych warstw:

warstwa 1: wejście/wyjście bota,

a) **Area Awareness System** – wejście, dostarcza wszystkich niezbędnych informacji o stanie świata otaczającego bota. Informacja o świecie gry jest otrzymywana jako zbiór danych bezpośrednio od programu gry. Wszystko, co bot jest w stanie zbadać za pomocą swoich zmysłów przechodzi przez tę warstwę.

b) **Basic Actions** – wyjście (analogiczne do użycia klawiatury i myszy dla gracza-człowieka), odpowiedzialne za wykonywanie podstawowych ruchów.

warstwa 2: określana jako inteligencja, która tkwi w podświadomości każdego człowieka, bot używa metod sztucznej inteligencji do określania celów(logika rozmyta), poruszanie się ścieżką ku celu. W tej warstwie znajduje się także moduł chat (interpreter i kreator wypowiedzi), tutaj przechowywane są również charakterystyczne cechy bota.

warstwa 3: odpowiedzialna za zachowanie w walce, omijanie przeszkód. Tutaj znajduje się także sieć bardzo podobna do maszyny stanów odpowiedzialna za wysokopoziomowe zachowanie bota. W warstwie tej mieści się także moduł komend – umiejętność odczytywania i rozumienia komend od lidera grupy.

warstwa 4: dodatkowa funkcjonalność lidera grupy, odpowiedzialna za organizowanie grupy i wykonywanie zadań grupowych.

5. Opis poszczególnych modułów architektury bota

AAS – Area Awareness System

System ten zapewnia botowi wszelkie niezbędne informacje na temat obecnego stanu świata gry. Dane te zawierają informacje o terenie oraz innych bytach w grze. Informacja jest pierwotnie przetwarzana by umożliwić szybki późniejszy dostęp do zinterpretowanych już częściowo danych.

Szkieletem ASS jest specjalna reprezentacja świata 3D. Wszystkie informacje dostarczane botowi są w jakiś sposób powiązane z tą reprezentacją. AAS

posiada podobną reprezentację jak system punktów kontrolnych – **ograniczone strefy**. Teren gry podzielony jest na strefy. Zakłada się, że koszt przemieszczenia się wewnątrz strefy jest minimalny(można przemieścić się - iść lub płynąć - w linii prostej).
Aby bot mógł poruszać się po całej strefie w linii prostej, obszar musi być wypukły. Wypukłe obszary nie mają w sobie przeszkód, które utrudniają poruszanie się. Jeśli bot ma przepłynąć przez strefę, wymaganie to jest wystarczające, jeśli jednak ma przez nią przejść, potrzebne są dodatkowe założenia. Obszar taki może bowiem mieć np. szczeliny w podłodze, do których wpaść może bot. Tak więc każdy obszar wypukły jest nawigowalny lub w łatwy sposób może być podzielony na obszary nawigowalne.

Dostępność strefy A ze strefy B oznacza, że istnieje możliwość bezpośredniego przemieszczenia się ze strefy B do strefy A. Najczęściej strefy są do siebie przyległe i nie ma dużego problemu obliczeniowego dostępności stref.

Dostępność stref jest możliwa poprzez:
- płynięcie w linii prostej
- przejście w linii prostej
- przejście w linii prostej „kucając”(*ang. crouching*)
- skakanie
- wyskakiwanie z wody
- teleportowanie
- użycie windy
- użycie skoczni
- „skok na rakiecie”(*ang. rocketjump*)

Oblicza się dostępność pomiędzy strefami, pasujące strefy są następnie odpowiednio łączone – zachowywana jest struktura połączeń pomiędzy strefami, która jest następnie używana przy poruszaniu się i wyszukiwaniu ścieżek przez bota.

Odnajdywanie ścieżek

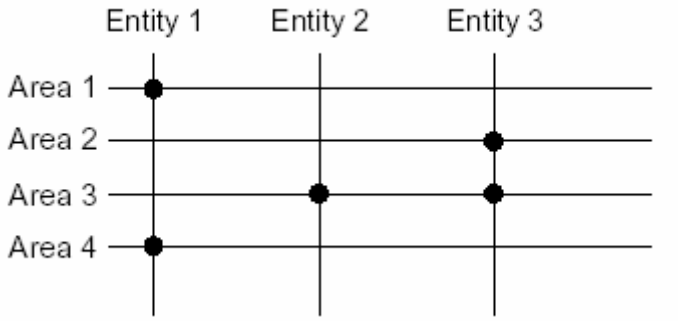
Posiadając informacje o dostępności stref, można obliczyć ścieżki pomiędzy strefami.

Istnieje możliwość użycia raz obliczonej tablicy wszystkich ścieżek i sprawdzenia w działającym programie jedynie danych w tej tablicy (może to pochłaniać sporo pamięci, jest jednak bardzo szybkie). Teren gry czasem się zmienia i trzeba dokonywać zmian w obliczonych ścieżkach. Takie rozwiązanie ma właśnie Quake 3. Posiada tymczasowy bufor z przekalkulowanymi ścieżkami, który zmienia się, gdy nastąpi zmiana na planszy (sytuacja taka zachodzi jednak dość rzadko).

Duże mapy mogą zawierać nawet 5000 stref. Algorytmy takie jak Dijkstra czy A* są zbyt czasochłonne do używania w czasie rzeczywistym. W Quake 3 użyty został algorytm wielopoziomowy. Najpierw wyszukiwana jest droga na wyższym poziomie, następnie droga do niższego bloku wewnątrz elementu wyższego poziomu.

Przedmioty

Przedmioty, które mogą być zebrane przez gracza są dzielone na grupy: zbroja lekka, zbroja ciężka itp. Dla każdej z grup może istnieć wiele instancji przedmiotów w świecie gry. Każdy z takich elementów ma przyporządkowaną dla siebie strefę – wewnątrz strefy odnalezienie przedmiotu nie jest już problemem. Łatwo jest np. ustalać ścieżkę taką, by po drodze zebrać jak najwięcej potrzebnych przedmiotów.



Rysunek 4. Przykładowy rozkład przedmiotów w strefach
Źródło: [1]

Basic Actions - Podstawowe akcje

Dla człowieka podstawowe akcje kontrolowania bohatera wykonywane są za pomocą klawiatury i myszy. Identycznymi akcjami dysponuje bot. Jednak akcje człowieka są ograniczone naturą fizyczną: wciśnięcie klawisza lub poruszenie myszą zajmuje pewną ilość czasu, dla bota brak tego ograniczenia. Jeśli chcielibyśmy uwzględnić ten element, należałoby funkcjonalność osobno zaimplementować.

Podstawowe akcje obejmują:

- atak – wystrzelenie z broni, którą postać obecnie posiada
- użycie – postać użyje przedmiotu, który trzyma
- „zmartwychwstanie" – bot powraca do życia
- skok
- kucnięcie
- poruszaj się w górę(pływając)
- poruszaj się w dół(pływając)
- poruszaj się wprzód
- poruszaj się w tył
- idź – postać będzie chodzić zamiast biegać
- rozmawiaj – pojawia się ikona nad postacią, postać nie może się poruszać
- gestykuluj – pokazanie gestu postacią bohatera
- poruszaj się(kierunek, prędkość)
- rozglądaj się
- wybierz broń
- mów – wysłanie komunikatu do wybranego gracza
- mów do drużyny – wysłanie komunikatu do całej drużyny

Charakter

Aby uczynić grę ciekawszą w Quake III wprowadzono wiele botów – każdy grający na swój sposób. Charakter każdego bota składa się ze starannie dobranych cech charakterystycznych bota.

Do cech charakterystycznych botów zalicza się między innymi:
- czujność,
- agresywność,

- mściwość,
- ciągły ogień podczas walki,
- celność(wiele aspektów),
- czas reakcji,
- preferencje broni,
- szybkość pisania(chat),
- skłonność do skoków/kucania

Większość charakterystyk przyjmuje wartości w zakresie [0,1]. Im wyższa wartość tym silniejsza i lepiej wykształcona cecha. Duża większość cech odnosi się do stylu walki bota – większość czasu gracz obserwuje bota właśnie w trakcie walki. Boty nie zmieniają swojego charakteru w trakcie gry. Każdy z botów ma na stałe przypisane cechy i zawsze zachowuje się zgodnie z nimi.

Fuzzy logic – logika rozmyta

Bot wykorzystuje logikę rozmytą do określania, jak bardzo chce wykonać daną czynność lub zdobyć dany przedmiot. Działanie logiki rozmytej bazuje na danych dostarczonych ze świata zewnętrznego np. jak daleko bot znajduje się od pożądanego przedmiotu, czy też ile ma obecnie energii. Wartość logiki rozmytej wyliczana jest dla wszystkich broni i przedmiotów. Bot najbardziej pożąda tego przedmiotu, dla którego wyliczona wartość jest największa.

Moduł logiki rozmytej wykorzystywany jest także do wyboru przez bota podczas walki najlepszej w danej sytuacji broni.

Dodatkowo każdy bot ma zależne od charakteru preferencje, co do używanej broni, czy zbieranych przedmiotów. Preferencje te są również wykorzystywane przy obliczaniu wartości rozmytych.

Chat

Bot musi mieć możliwość komunikowania się z innymi graczami w grze (także z botami). Bot musi interpretować otrzymywane wiadomości tekstowe od innych graczy oraz mieć zdolność do tworzenia i wysyłania takich wiadomości.

W Quake 3 interpretacja wiadomości tekstowych polega na odnajdywaniu słów kluczowych w tekście. Wyniki są porównywane z predefiniowanymi wzorcami. Nie zastosowano złożonego mechanizmu parse'owania i głębszej analizy tekstu, gdyż takie rozwiązania są zazwyczaj czasochłonne, a to nie jest dopuszczalne w grze komputerowej. Ponadto w pisanych przez graczy tekstach często zdarzają się błędy, używanie skrótów, czy przypadkowe pominięcia liter.

Silnik przetwarzania tekstu posiada zbiór słów oraz synonimów każdego słowa. Silnik zawiera także zbiór predefiniowanych wypowiedzi, które są używane z różnych sytuacjach, również dla każdej zdefiniowanej sentencji wypowiadanych przez innego gracza posiada zbiór sentencji, które może użyć jako odpowiedź.

Przykładowym zdefiniowanym fragmentem rozmowy może być[1]:

```
["hate you", !"not"] = 7
{
"why do you hate me";
"there's no reason for you to hate me";
}
```

Jeśli bot wyszuka w wiadomości gracza ciąg słów „hate you" oraz nie znajdzie słowa „not", będzie mógł odpowiedzieć na dwa sposoby.

Silnik chatu w grze Quake 3 posiada jeszcze kilka dodatkowych wariancji rozmów, które zostały tutaj pominięte. Dokładniejsze informacje można znaleźć w [1].

Goals – cele botów

W grze wyodrębniamy dwa typy celów: cele długoterminowe i cele krótkoterminowe. Najważniejszym celem w grze jest oczywiście odniesienie szeroko pojmowanego zwycięstwa. Podczas próby osiągnięcia pewnego długoterminowego celu bot przemieszczając się z jednego miejsca na planszy w drugie, wyznacza sobie cele krótkoterminowe np. zdobycie lepszej broni, które jednak nie przeszkadzają zbytnio w osiąganiu celu długoterminowego.

Często jednym z celów długoterminowych jest zdobycie lepszej broni, czy zbroi, gdyż znacznie podwyższa to szanse bota na zwycięstwo. Po osiągnięciu celu długoterminowego bot decyduje się na kolejny cel (używany jest moduł logiki rozmytej). Bot może zdecydować się na atak na wroga bądź też przykładowo zdobycie lepszych przedmiotów.

Navigation – przemieszczanie się

Opisany wcześniej AAS dostarcza wszelkich informacji dla bota niezbędnych do poruszania się po planszy. Bot używa dwóch sposobów nawigacji: nawigacja w pewnym kierunku oraz nawigacja do określonego celu.

Nawigacja do celu

Jeśli bot znajduje się w strefie, w której znajduje się również cel, wystarczy przemieszczać się w linii prostej do celu. W przeciwnym przypadku, system wyznaczania ścieżek określa następną strefę, do które bot musi się przemieścić, by dotrzeć do celu. Informacje dostarczone od tego modułu nie są jednak bezwzględnie wykonywane, gdyż bot może zdecydować na ominięcie pewnej przeszkody i „zahaczenie" o przyległą strefę. Dla każdego ze sposobów dostępności stref bot posiada zaimplementowany minimoduł odpowiedzialny za efektywne przemieszczanie się pomiędzy strefami. Tak poruszając się, bot będzie przechodził pomiędzy strefami, aż dotrze do strefy docelowej.

Nawigacja w określonym kierunku

Bot przemieszczając się w określonym kierunku nie używa systemu nawigacji do wyznaczania kolejnych stref, do których powinien się przemieszczać. Bot wykorzystuje natomiast próbkowanie otoczenia. Bot porusza się w danym kierunku, jeśli przykładowo natrafi na podest, spróbuje na niego wskoczyć, jeśli trafi na szczelinę w podłodze, wskoczy w nią pod warunkiem, że nie spadnie w przepaść lub nie wyląduje w lawie.

Misc. AI

Moduł ten jest używany w sytuacjach, gdy oczekuje się inteligentnego zachowania od bota w danym momencie np. podczas walki.

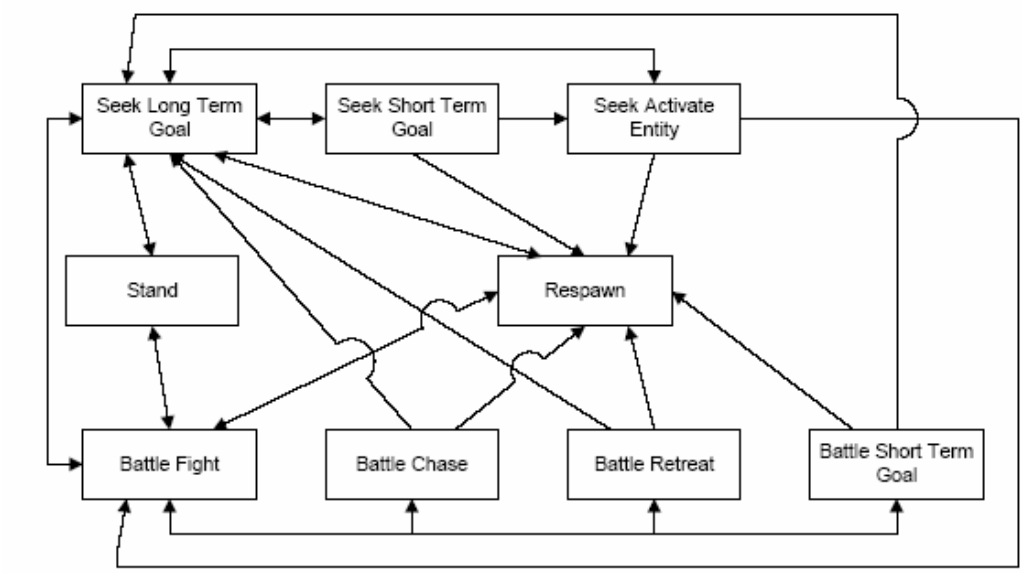
Przykładowo, jeśli to przeciwnik pierwszy spostrzegł bota i strzelił do niego, bot rozejrzy się w koło i sprawdzi skąd dochodzą strzały. Kąt widzenia bota ograniczony jest do 90 stopni, tak by imitować dokładniej zachowanie człowieka. Widoczność bota jest także ograniczana przez mgłę oraz przez artefakty zapewniające przeciwnikom niewidoczność.

Również podczas samej walki moduł ten odpowiedzialny jest za wiele elementów. Bot posiada określoną precyzję celowania (zależną od charakteru), nie byłoby dobrym rozwiązaniem, gdyby bot był perfekcyjnym strzelcem. Bot nie powinien stać podczas walki w jednej pozycji, poruszając się w odpowiedni sposób uniknie znacznej liczby trafień. Strzelając z broni, której pociski mają większe opóźnienie (np. rakietnice) bot w odpowiedni sposób przewiduje pozycję gracza w momencie, gdy pocisk będzie się do niego zbliżał i wystrzeli go właśnie w tym kierunku.

Wszystkie te zachowania bota zostały odpowiednio zaprogramowane i umieszczone w module Misc. AI. Znacznie trudniej byłoby sprawić, że podczas gry bot sam nauczyłby się odpowiedniego zachowania, lecz takie rozwiązania są także stosowane.

AI network

Moduł ten określany jest jako centralny mózg bota. Sieć ta przypomina maszynę ze skończoną ilością stanów, jest modelowana jako sieć stanów połączonych warunkowymi przejściami. Zawsze istnieje jeden najlepszy stan w danej sytuacji.



Rysunek 5. Sieć AI
Źródło: [1]

Sieć składa się z następujących stanów:

a) **Respawn** – bot wchodzi w ten stan, gdy umiera, w stanie tym bot ma możliwość odrodzenia się,
b) **Stand** – bot stoi najczęściej, gdy używa modułu chat,
c) **Seek long term goal** – bot próbuje osiągnąć cel długoterminowy,
d) **Seek short term goal** – bot próbuje osiągnąć cel krótkoterminowy,
e) **Seek active entity** – bot próbuje rozwiązać problem podczas dążenia do celu np. strzelenie w przycisk, aby otworzyły się drzwi,
f) **Battle fight** – bot walczy wrogiem,
g) **Battle chase** – bot ściga przeciwnika,
h) **Battle retreat** – bot wycofuje się z walki,
i) **Intermission** – bot przechodzi do tego stanu po zakończeniu gry
j) **Observer** – bot przechodzi do tego stanu, gdy zginie w trybie obserwatora

Commands – komendy wydawane graczom

Moduł komend odpowiedzialny jest za odbieranie i interpretację komend pochodzących od członków drużyny. Komunikacja odbywa się poprzez chat, bot jednak musi być w stanie odpowiednio zareagować.

Przykładowo może otrzymać jedno z poleceń:
- pomóż innemu graczowi
- obroń pewną strefę
- zdobądź flagę przeciwnika
- wracaj do bazy
- patroluj

Bot zawsze stara się wypełniać polecenie lidera grupy, chyba że nie pozwala na to bieżąca sytuacja np. bot jest atakowany.

Dostępnych poleceń oczywiście jest znacznie więcej. Moduł komend znacznie usprawnia drużynową grę i pozwala na lepszą interakcję z innymi graczami. Każdy bot umie także odpowiedzieć na zadawane mu pytania, przykładowo:
- gdzie jesteś?
- jakie zadanie obecnie wykonujesz?
- w której drużynie jesteś?
- co powinienem robic?

Team AI – inteligencja dowódcy drużyny

Moduł komend nie może być w pełni wykorzystany, jeśli brak dowódcy drużyny. Musi istnieć osoba(zwłaszcza w grach bardziej zaawansowanych, np. CTF), która będzie decydować o strategii i wydawać polecenia innym członkom drużyny. Moduł inteligencji takiego bota to właśnie Team AI.

Podsumowanie

W niniejszej pracy przedstawiono zagadnienie botów w grach komputerowych 3D skupiając się głównie na metodzie ich tworzenia, architekturze oraz sposobie funkcjonowania. Temat inteligencji botów jest bardzo rozległy i obejmuje wiele dziedzin sztucznej inteligencji. Jest to jednak temat bardzo ciekawy oraz popularny i na pewno warty zainteresowania.

Bibliografia

[1] http://www.kbs.twi.tudelft.nl/docs/MSc/2001/Waveren_Jean-Paul_van/thesis.pdf
[2] Game Programing Gems: Bruno Miguel Teixeira de Sousa
[3] http://ai-depot.com/GameAI/Bot-Introduction.html
[4] http://theory.stanford.edu/~amitp/GameProgramming/
[5] http://mag.dsi.unimi.it/qsmodels/level4/
[6] http://www.cgf-ai.com/docs/straatman_remco_killzone_ai.pdf
[7] http://www.telefragged.com/thefatal/old/genebot.shtml
[8] http://www.microconsultants.com/tips/fsm/fsmarticl.pdf
[9] http://graphics.stanford.edu/~quake/resources.html
[10] http://www.planetquake.com/minion/tutorial%5Cmain.htm