Rozdział 3.5

Robot sterowany celem: projektowanie zachowań i cech osobowości agenta gry przy użyciu rozszerzonych sieci behawioralnych

Hugo Pinto i Luis Otavio Álvares
hugo@hugopinto.net

Przy projektowaniu robota będącego aktorem gry (zwanego powszechnie *botem*[1]) podstawowym problemem jest symulowanie zachowania podporządkowanego określonemu celowi (celom) i wynikający stąd problem selekcji odpowiednich akcji. Zadanie to staje się szczególnie skomplikowane, gdy cele te są wzajemnie sprzeczne; jeżeli w dodatku robotowi przychodzi działać w dynamicznie zmieniającym się środowisku i musi na bieżąco uwzględniać mnóstwo czynników, stajemy — my, projektanci — przed naprawdę trudnym problemem. Tradycyjne podejście oparte na przeglądaniu bazy wiedzy okazuje się niepraktyczne ze względu na ogromny rozmiar przestrzeni przeszukiwania, a starannie opracowana akcja może okazać się chybiona, bowiem w trakcie jej planowania mogły drastycznie zmienić się uwarunkowania środowiskowe przyjęte jako przesłanki podczas pisania planów.

Jednym z najczęściej stosowanych mechanizmów selekcyjnych, sprawdzających się w złożonych i zmiennych środowiskach, są sieci behawioralne, nieustannie ewoluujące od momentu pierwszego zaistnienia w literaturze [Maes89], czego przykładem są chociażby prace [Tyrrell93], [Rhodes96], [Goetz97], [Dorer99] i [Nebel03]. Sieci behawioralne

[1] Patrz *http://pl.wikipedia.org/wiki/Bot_%28program%29 — przyp. tłum.*

znalazły zastosowanie m.in. w symulacji zachowania zwierząt [Tyrrell93], w interaktywnych opowiadaniach (*interactive storytelling*) [Rhodes96], projekcie RoboCup [Müller01] i grze *Unreal Tournament* [Pinto05-a].

Rozszerzone sieci behawioralne [Dorer04] reprezentują aktualny „stan sztuki" wśród sieci behawioralnych. Sieci te bazują na ciągłym wartościowaniu (za pomocą liczb rzeczywistych) warunków wstępnych i efektów, pozwalając na specyfikowanie celów zależnych od aktualnej sytuacji i współbieżną selekcję akcji.

Właściwy wybór akcji to jednak nie wszystko: w grze komputerowej nie mniej istotnym zagadnieniem jest sposób tego wyboru oraz to, jak powiązany jest on z (symulowanymi) cechami osobowości agenta. Ta interesująca koncepcja doczekała się praktycznej realizacji — w pracy magisterskiej Bradleya Rhodesa [Rhodes96] opisany jest model sieci behawioralnej PHISH-NET do projektowania symulacji cech osobowości, a my wykrystaliśmy rozszerzoną sieć behawioralną do projektowania stereotypów w grze *Unreal Tournament* [Pinto05-b].

Wszelkie „strzelanki" typu FPP — jak *Unreal Tournament* — to interesująca domena zastosowań rozszerzonych sieci behawioralnych. Agent usytuowany w ciągłym, trójwymiarowym środowisku nieustannie oddziaływuje (w rozmaity sposób) z wieloma encjami w tymże środowisku, według zmiennych i złożonych scenariuszy. Agent ów ma do dyspozycji wiele rodzajów broni, o odmiennej charakterystyce; porusza się w środowisku zróżnicowanych krajobrazów i wchodzi w interakcję z kilkoma innymi agentami. Szeroki repertuar wykonywanych przez niego akcji obejmuje (m.in.) bieg, spacer, zwroty, pływanie, strzelanie i ostrzeliwanie, zmianę broni, skoki, pobieranie obiektów i korzystanie z nich, a wiele z tych akcji wykonywanych może być współbieżnie (np. bieg i strzelanie). Agent musi też umiejętnie godzić ze sobą sprzeczne cele, takie jak ofensywność i własne bezpieczeństwo.

Większość występujących w grze postaci to charaktery raczej powierzchowne i stereotypowe, modelowanie ich cech osobowych okazuje się w pełni wystarczające. W niniejszym rozdziale pokażemy jednak wyniki własnych eksperymentów w zakresie implementowania różnych profili osobowych przy użyciu rozszerzonych sieci behawioralnych, czyli dostosowywania i oceniania rozmaitych strategii wyboru wykonywanych akcji.

Rozszerzone sieci behawioralne

Rozszerzona sieć behawioralna (ang. *Extended Behavior Network — EBN*) może być rozpatrywana jako zbiór połączonych modułów i celów, wzajemnie się wzbudzających lub powstrzymujących za pomocą dystrybuowania i rozpraszania energii aktywacyjnej — poczynając od celów i przepływając przez poszczególne moduły. W każdym kroku symulacji do wykonywania wybierane są moduły posiadające najwyższy zasób energii i najwyższą „wykonywalność", pod warunkiem dostępności niezbędnych do tego zasobów. W dalszej części rozdziału rozpatrzymy szczegółowo strukturę przykładowej sieci EBN oraz przykładowy algorytm wyboru akcji do wykonywania przez agenta.

Struktura

Struktura sieci EBN to kolekcja predefiniowanych zbiorów: modułów behawioralnych, celów, połączeń modułów z celami i innymi modułami, zasobów i parametrów sterujących. Na rysunku 3.5.1 przedstawiona jest specyfikacja prostej sieci wykorzystywanej w naszych eksperymentach, zaś na rysunku 3.5.2 widoczny jest schemat sieci zbudowanej w oparciu o tę specyfikację: prostokąty z zaokrąglonymi narożnikami reprezentują cele, zwykłe prostokąty — moduły *behawioralne*, natomiast ośmiokąty symbolizują *zasoby*. Linie ciągłe to *połączenia wsteczne* (ang. *predecessor links*)[2], linie przerywane to *połączenia kolizyjne* (ang. *conflict links*), zaś uzależnienia poszczególnych modułów od zasobów niezbędnych do ich wykonywania oznaczone są liniami kropkowanymi.

MODUŁ
Nazwa: *ShootEnemy*
Warunek wstępny: *EnemyInSight*
Akcja: *ShootEnemy*
Efekty: *EnemyHurt* 0,6
LowAmmo 0,1
KONIEC MODUŁU

ZASÓB
Nazwa: *Nogi*
Ilość: *2*
KONIEC ZASOBU

ZASÓB
Nazwa: *Głowa*
Ilość: *1*
KONIEC ZASOBU

ZASÓB
Nazwa: *Ręce*
Ilość: *2*
KONIEC ZASOBU

CEL
Nazwa: *EnemyHurt*
Kontekst: *(brak)*
Warunek: *EnemyHurt*
Potencjał: *1,0*
KONIEC CELU

CEL
Nazwa: *Not LowAmmo*
Kontekst: *LowAmmo*
Warunek: *Not LowAmmo*
Potencjał: *0,7*
KONIEC CELU

PARAMETRY
Nazwa: *ActivationInfluence* **Wartość:** *1,0*
Nazwa: *InhibitionInfluence* **Wartość:** *0,9*
Nazwa: *Inertia* **Wartość:** *0,5*
Nazwa: *GlobalThreshold* **Wartość:** *0,6*
Nazwa: *ThresholdDecay* **Wartość:** *0,1*
KONIEC PARAMETRÓW

Rysunek 3.5.1. *Specyfikacja prostej sieci EBN*

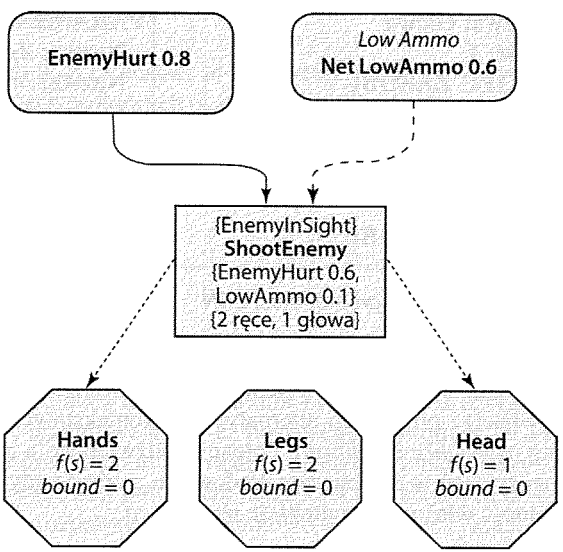
Cel opisywany jest w postaci trzech elementów: propozycji pewnego warunku, którego spełnienia się oczekuje, potencjału (ang. *strength* — wyrażanego w postaci liczby rzeczywistej) i propozycji alternatywnej (kontekstowej), zwanej *warunkiem adekwatności* (ang. *relevance condition*). Potencjał odzwierciedla istotność celu w oderwaniu od aktualnej sytuacji w otaczającym środowisku, podczas gdy warunek adekwatności wyraża jego istotność właśnie w kontekście tej sytuacji.

[2] Patrz rozdział 3.3 niniejszej książki — *przyp. tłum.*

Wyrażenie istotności celu jako pary dwóch wartości — statycznej i dynamicznej — daje lepsze możliwości dostosowywania miary celu do aktualnej sytuacji i możliwości agenta; na rysunku 3.5.1 przykładem takiego celu jest `EnemyHurt`. Zauważmy, że cel o określonego warunku adekwatności to cel zawsze adekwatny, czyli cel o maksymalnej wartości.

Każdy moduł w rozszerzonej sieci behawioralnej stanowi kombinację listy warunków wstępnych, akcji, listy efektów i listy zasobów. Pierwsza ze wspomnianych list stanowi koniunkcję warunków wstępnych niezbędnych do wykonania modułu; lista efektów to koniunkcja warunków (będących zwykle negacjami), jakich spełnienia oczekujemy w wyniku wykonania modułu. Elementy listy zasobów mają natomiast postać par (zasób, ilość) odzwierciedlających ilości poszczególnych zasobów, jakich agent potrzebuje do swego działania.

Moduły behawioralne i cele mogą być połączone w dwojaki sposób. Połączenia wstępne to połączenia skierowane od modułu lub celu B do modułu A, wiążące elementy wstępne listy efektów modułu A z listą warunków wstępnych modułu B, przy czym łączone elementy mają tę samą wartość logiczną — `true` (+) albo `false` (-). Na rysunku 3.5.2 przykładem takiego połączenia jest połączenie celu `EnemyHurt` z modułem `ShootEnemy`. Analogiczne połączenie między elementami o różnej wartości logicznej nazywa się połączeniem *kolizyjnym* (ang. *conflict link*); przykładem takiego połączenia na rysunku 3.5.2 jest połączenie celu `Not LowAmmo` z modułem `ShootEnemy`. Połączenia kolizyjne reprezentują wyczerpywanie energii ze swych modułów docelowych, podczas gdy połączenia wsteczne — przeciwnie — wzbogacają energię swych modułów docelowych. W ten sposób dany moduł lub cel może zwiększać lub zmniejszać szansę wykonywania poszczególnych modułów, zależnie od warunków zachodzących w środowisku zewnętrznym.



Rysunek 3.5.2. *Diagram prostej sieci EBN*

Każdy zasób w sieci behawioralnej reprezentowany jest przez węzeł, z którym stowarzyszona jest funkcja $f(s)$ określająca (spodziewaną) ilość tegoż zasobu dostępną w sytuacji s. Ponadto z węzłem tym stowarzyszone są dwie inne wielkości: wykorzystywana

aktualnie ilość zasobu (reprezentowana przez zmienną *bound*) i tzw. próg aktywowania zasobu θ_{Res} będący dodatnią liczbą rzeczywistą nie większą niż *globalny* próg aktywacji θ. Na rysunku 3.5.2 funkcja f jest stała dla każdego zasobu — nic dziwnego, wszak robot ma zawsze dwie nogi, dwie ręce i jedną głowę (nie przewidujemy żadnych makabrycznych przygód w rodzaju dekapitacji czy amputacji kończyn).

Jak wcześniej wspominaliśmy, uzależnienia modułów behawioralnych od zasobów reprezentowane są na diagramach za pomocą linii kropkowanych; ponadto dla każdego modułu specyfikowana jest ilość każdego z zasobów niezbędna do jego wykonania.

Do ostatecznego dostrajania sieci służą parametry sterujące, przyjmujące wartości z przedziału [0, 1]. Parametr γ, zwany *podatnością aktywacyjną* (ang. *activation influence*), kontroluje ilość energii aktywacyjnej dostarczanej przez łącza wsteczne; analogicznie parametr δ, zwany *podatnością tłumienia* (ang. *inhibition influence*), równy jest zanegowanej wartości współczynnika tłumienia wprowadzanego przez połączenia kolizyjne. Znaczenie pozostałych parametrów — bezwładności (ang. *inertia*) β, globalnego progu aktywacji (ang. *global threshold*) θ i szybkości zaniku tego progu w czasie (ang. *threshold decay*) $\Delta\theta$ — wyjaśnimy w dalszym ciągu rozdziału.

Algorytm wyboru akcji

Wybór modułu behawioralnego do wykonania w każdym cyklu funkcjonowania sieci odbywa się według następującego scenariusza.

1. Dla każdego modułu wyliczana jest wartość energii aktywacji a.
2. Dla każdego modułu obliczana jest wartość tzw. wykonywalności (ang. *executability*) e jako wartość dowolnej normy trójkątnej[3] jego warunków wstępnych.
3. Dla każdego modułu obliczany jest iloczyn $h(a, e) = a*e$, odzwierciedlający kombinację energii aktywacji i prawdopodobieństwa pomyślnego wykonania (wykonywalności). Dzięki temu nawet moduły z bardzo słabo spełnionymi warunkami wstępnymi mogą zostać wybrane do wykonania, o ile mają wystarczająco dużą wartość aktywacji.

[3] Normą trójkątną (ang. *triangular norm*), w skrócie *t-normą*, nazywamy funkcję dwuargumentową T o wartościach argumentów z przedziału [0, 1] i wyniku z tegoż przedziału, spełniającą następujące warunki:

1. Przemienność — dla każdego $a, b \in [0, 1]$, $T(a, b) = T(b, a)$,
2. Łączność — dla każdego $a, b, c \in [0, 1]$, $T(T(a, b), c) = T(a, T(b, c))$,
3. Monotoniczność — dla każdego $a, b, c, d \in [0, 1]$, $a \le c \wedge b \le d \Rightarrow T(a, b) \le T(c, d)$,
4. Warunki brzegowe — dla każdego $a \in [0, 1]$, $T(a, 0) = 0 \wedge T(a, 1) = a$.

Przykładem normy trójkątnej jest funkcja wybierająca mniejszą z dwóch wartości oraz zwykły iloczyn arytmetyczny. Normę trójkątną uogólnić można na dowolną liczbę argumentów, wykorzystując własność łączności w następujący sposób:

$$T*(a_1, a_2, a_3, \ldots a_n) = T(a_1, T(a_2, T(a_3, T(\ldots T(a_{n-1}, a_n))\ldots)))$$

i w tym właśnie sensie jest ona stosowana do obliczania wykonywalności modułu na podstawie listy jego warunków wstępnych — *przyp. tłum.*

4. Dla każdego zasobu wykorzystywanego przez dany moduł, poczynając od ostatniego niedostępnego zasobu, sprawdzany jest warunek $a \geq \theta_{Res}$ oraz dostępność w ilości (co najmniej) wymaganej przez moduł. Jeśli te dwa warunki są spełnione, zasób jest wiązany z modułem.
5. Jeżeli dla danego modułu uda się powiązać wszystkie wymagane przezeń zasoby, moduł ten zostaje wybrany do wykonania, jednocześnie próg aktywacji θ_{Res} każdego z powiązanych zasobów resetowany jest do wartości globalnej θ.
6. Po zakończeniu wykonywania modułu wykorzystywane przez niego zasoby są zwalniane.

Próg aktywacji θ_{Res} każdego zasobu zmniejsza się liniowo w czasie, co daje kandydującemu modułowi szansę wykonania, proporcjonalną do jego wartości aktywacyjnej a.

Wzór (3.5.1) określa energię aktywacji przepływającą w kroku t przez łącze wstępujące od celu g_i do modułu k. Funkcja f jest dowolną normą trójkątną obliczającą wynik kombinacji potencjału celu i jego warunku adekwatności. Czynnik ex_j jest wartością energii na drugim końcu łącza.

$$a_{kg_i}^{t\ (1)} = \gamma * f\left(\lambda_g, r_{g_i}^t\right) * ex_j \tag{3.5.1}$$

Ubytek energii aktywacyjnej w kroku t poprzez łącze kolizyjne między celem g_i a modułem k określony jest wzorem:

$$a_{kg_i}^{t\ (2)} = -\delta * f\left(\lambda_{g_i}, r_{g_i}^t\right) * ex_j \tag{3.5.2}$$

Ilość energii aktywacyjnej przepływającej w kroku t od modułu *succ* do modułu k (przez łącze wsteczne) określona jest wzorem:

$$a_{kg_i}^{t\ (3)} = \gamma * \sigma\left(a_{succ\,g_i}^{t-1}\right) * ex_j * \left(1 - \tau\left(p_{succ}, s\right)\right) \tag{3.5.3}$$

p_{succ} oznacza wartość propozycji modułu *succ*, zaś a_{succ} jest jego energią aktywacji. $\tau(p_{succ}, s)$ oznacza wartość p_{succ} w sytuacji s. Jak widać, przepływ energii aktywacyjnej między modułami jest tym większy, im słabiej spełniony jest warunek na początku łącza wstecznego między tymi modułami. W tym kontekście niespełnione warunki jawią się jako coraz bardziej pożądane podcele. Formuła (3.5.4) definiuje moduł behawioralny jako tzw. mocny atraktor (ang. *strong attractor*) [Goetz97] o wysokim prawdopodobieństwie. Redukuje to ilość niepotrzebnych przełączeń między modułami, ponieważ małe zmiany w obserwowanym środowisku z mniejszym prawdopodobieństwem powodować będą przerywanie rozpoczynanych właśnie akcji.

$$\sigma(x) = \left(1 + e^{k(\mu - x)}\right)^{-1} \tag{3.5.4}$$

Analogicznie do formuły (3.5.3), formuła (3.5.5) określa ilość energii przekazywanej w kroku t przez łącze *kolizyjne* między modułami; a_{conf} oraz p_{conf} oznaczają (odpowiednio) energię aktywacji i wartość propozycji w module początkowym tego łącza.

$$a_{kg_i}^{t\ (4)} = -\delta * \sigma\left(a_{conf\ g_i}^{t-1}\right) * ex_j * \tau\left(p_{conf}, s\right) \tag{3.5.5}$$

Zgodnie z formułą (3.5.6), energia aktywacji modułu k w kroku t równa jest jego energii aktywacyjnej w kroku poprzednim (t–1) przemnożonej przez stałą bezwładności β i zwiększonej o sumę aktywacji zachowywanej w każdym z celów g_i.

$$a_k^{t\ (1)} = \beta\ a_k^{t-1} + \sum_i a_{kg_i}^t \tag{3.5.6}$$

Ostatecznie, zgodnie z formułą (3.5.7), każdy moduł zachowuje energię aktywacji o największej wartości bezwzględnej z każdego celu. Jest to równoważne zachowywaniu tylko najtrwalszych ścieżek od wspomnianego modułu do każdego z celów.

$$a_{kg_i}^t = \mathrm{abs}\left(\max\left(a_{kg_i}^{t\ (1)}, a_{kg_i}^{t\ (2)}, a_{kg_i}^{t\ (3)}, a_{kg_i}^{t\ (4)}\right)\right) \tag{3.5.7}$$

Jakość wyboru akcji

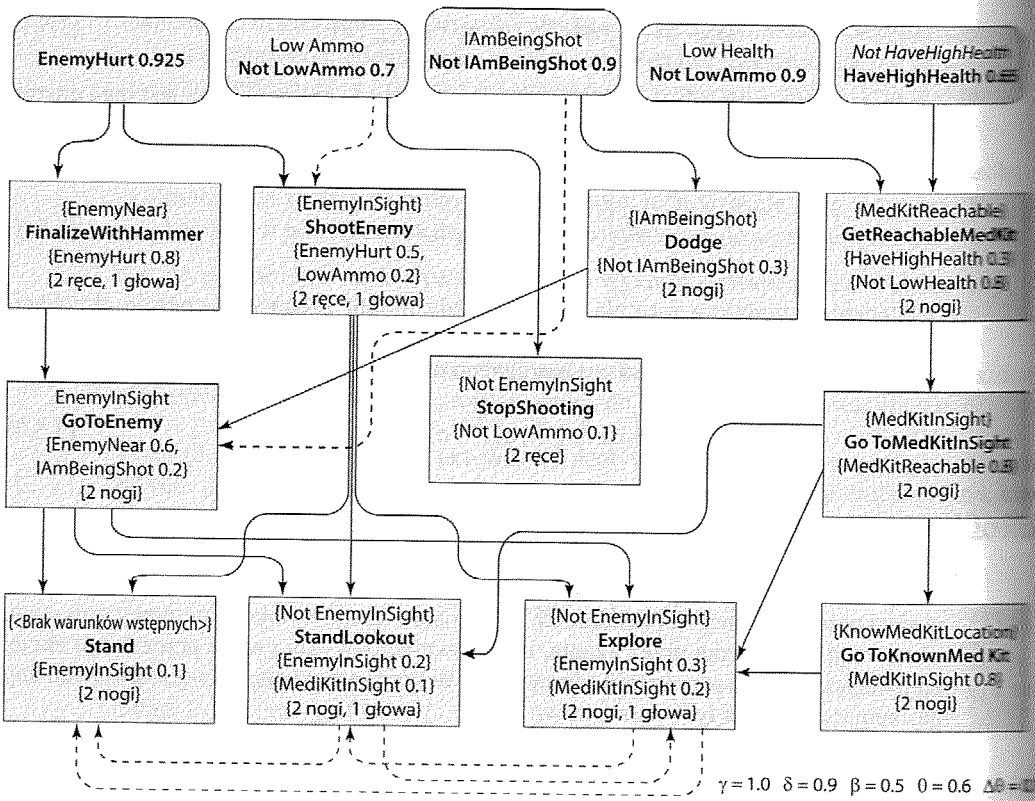
Przeprowadziliśmy szereg eksperymentów w celu oceny jakości algorytmu wyboru akcji w sieci przedstawionej na rysunku 3.5.3. Zastosowaliśmy wysokopoziomowy opis stanu obserwacji środowiska przez agenta, rejestrowaliśmy wykonywane przez niego akcje, zmieniając wartości parametrów sterujących. W konfiguracji domyślnej zastosowaliśmy następujące wartości tych parametrów:

- γ (*ActivationInfluence*) = 1,0
- δ (*InhibitionInfluence*) = 0,9
- β (*Inertia*) = 0,5
- θ (*GlobalThreshold*) = 0,6

W większości przypadków wartości te okazywały się odpowiednie, jednak w kilku eksperymentach użyliśmy ich wartości ekstremalnych (szczególnie w przypadku parametrów β oraz δ). W dalszym ciągu dyskusji opierać się będziemy na zapisach w kronice eksperymentu oraz bezpośrednich obserwacjach zachowania agenta.

Zachowanie agenta

Generalnie zachowanie agenta można opisać w następujący sposób. Rozpoczyna on od eksploracji terenu, wędrując w poszukiwaniu przeciwnika (`Explore`). Natrafiwszy na przeciwnika, rozpoczyna strzelanie (`ShootEnemy`) i próbuje zbliżyć się do niego (`GoToEnemy`), a gdy mu się to uda, zmienia broń na młot pneumatyczny jako bardziej skuteczny w walce bezpośredniej (`FinalizeWithHammer`). Gdy przeciwnik zostanie zabity, agent przerywa strzelanie (`StopShoot`) i rozpoczyna wędrówkę na nowo. Gdy agent zbliży się do przeciwnika, zatrzymuje się na chwilę i, odskakując, próbuje uniknąć kierowanych do niego strzałów. Gdy przeciwnik przestaje strzelać, agent ponownie zaczyna się posuwać w jego kierunku. Gdy agent zostanie zraniony i zna lokalizację któregoś z zestawów medycznych (warunek `KnownMedKit` (+) ma wartość bliską 1,0), korzysta z niego,



Rysunek 3.5.3. *Sieć behawioralna wykorzystywana w eksperymentach badających jakość wyboru akcji*

odzyskując kondycję po krótkiej chwili. Gdy kondycja agenta osiągnie bardzo niski poziom w czasie zbliżania się do przeciwnika, a w zasięgu ręki znajduje się zestaw medyczny (`MedKitReachable`), agent zatrzymuje się, korzysta z zestawu i nie przestaje strzelać — chyba że znajduje się bardzo blisko przeciwnika, wtedy zamiast karabinu używa młota pneumatycznego (`FinalizeWithHammer`).

Sekwencjonowanie akcji

Zaobserwowaliśmy tendencję do grupowania się wykonywanych przez agenta akcji w pewne ustalone sekwencje. Sekwencja {`StopShooting`, `Explore`, `GoToEnemy`, `ShootEnemy`, `FinalizeWithH`} okazywała się najczęściej występującą sekwencją ataku, zaś sekwencja {`GoToKnownMedKit`, `GoToMedKitInSight`, `GetReachableMedKit`} pojawiała się najczęściej w sytuacji, gdy w polu widzenia agenta nie znajdował się żaden przeciwnik, a kondycja agenta nie była najlepsza. Długa sekwencja {`Explore`, `GoToEnemy`, `ShootEnemy`, `FinalizeWithHammer`, `StopShooting`, `GoToKnownMedKit`, `GoToMedKitInSight`, `GetReachableMedKit`} — stanowiąca (w przybliżeniu) złożenie dwóch poprzednich — odpowiadała najczęściej całościowemu zachowaniu agenta. Zwróćmy uwagę, iż logiczny, długi łańcuch czynności zrealizowany został bez uprzedniego (formalnego) zaplanowania.

Reaktywność i konsekwencja

Jak widzieliśmy, sekwencja akcji {Explore, GoToEnemy, FinalizeWithHammer} stanowi „plan" spełnienia celu EnemyHurt. Gdy w trakcie zbliżania się do przeciwnika agent zostaje postrzelony, przerywa akcję GoToEnemy, „przełącza" się na akcję BehaviorDodge i po udanym uniknięciu kolejnych strzałów powraca do akcji GoToEnemy. Postępowanie takie jest wynikiem udanego kompromisu między konsekwencją a reaktywnością na zmiany zachodzące w środowisku. Im większa wartość bezwładności (parametr β), tym refleks agenta mniejszy: przy dużej wartości β agent reaguje odskokami jedynie na *serie* strzałów (a nie pojedyncze strzały).

Rozstrzyganie konfliktów

Rzut oka na rysunek 3.5.3 pozwala stwierdzić, że do spełnienia celu EnemyHurt konieczne jest wykonanie akcji ShootEnemy, która z kolei sprzeczna jest z celem Not LowAmmo. Ten ostatni jest mało istotny, dopóki agent ma pod dostatkiem amunicji; gdy ta zaczyna się wyczerpywać, agent kończy strzelanie i sięga po młot pneumatyczny (akcja FinalizeWithHammer), ten bowiem nie wykorzystuje żadnych zasobów wyczerpujących się. W ten nietypowy, lecz skuteczny sposób udaje się pogodzić sprzeczne cele (ofensywność i oszczędzanie amunicji); zauważmy, że „standardową" akcją prowadzącą do oszczędzania amunicji jest StopShooting.

Inny konflikt celów związany jest z akcją GoToEnemy. W dążeniu do jak najlepszego wypełnienia celu EnemyHurt agent próbuje zbliżyć się do wroga, ryzykując postrzelenie, co przeczy celowi Not IAmbeingShot. Aby te sprzeczne cele pogodzić ze sobą, agent zachowuje pewną odległość od przeciwnika. Sieć behawioralna radzi sobie z tym konfliktem dość dobrze: ostrzeliwany agent stosuje uniki, systematycznie (choć z przerwami) zbliżając się do przeciwnika.

Preferowanie akcji wielocelowych

W naszej sieci behawioralnej akcje StandLookout i Explore mają pierwszeństwo przed akcją Stand w sytuacji, gdy ich moduły mają identyczną wartość wykonywalności — nawet wówczas, gdy z akcją Stand związana jest większa wartość efektu EnemyInSight (czyli gdy — mówiąc po ludzku — wypatrywanie wroga bez ruchu jest bardziej skuteczne niż rozglądanie się wokoło czy eksplorowanie terenu). Powód tej preferencji jest prosty: zarówno StandLookout, jak i Explore wnoszą swój wkład w spełnienie dwóch celów — EnemyHurt i HaveHighHealth — podczas gdy Stand prowadzić może jedynie do spełnienia celu EnemyHurt (pośrednio, poprzez ShootEnemy). Generalnie — moduły związane z większą liczbą celów akumulują zwykle więcej energii aktywacyjnej i tym samym częściej wybierane są do wykonania.

Właściwe łączenie akcji współbieżnych

Nasz agent znakomicie radzi sobie z zarządzaniem dostępnymi zasobami i umiejętnie łączy ze sobą poszczególne akcje: strzela, unikając jednocześnie trafienia, czy też przestaje strzelać, gdy zbliża się do przeciwnika, eksploruje teren lub sięga po zestaw pierwszej

pomocy (choć może też sięgać po ów zestaw, nie przestając strzelać). Wszelkie kombinacje współbieżnych akcji okazują się sensowne; co więcej, udało nam się zaobserwować wszelkie sensowne ich kombinacje, jakie tylko mogliśmy sobie wyobrazić.

Przeprowadzone eksperymenty dowiodły niezbicie, że rozszerzone sieci behawioralne stanowią znakomite rozwiązanie w zakresie mechanizmów selekcji akcji dla sterowanego celem agenta gry. W dalszym ciągu niniejszego rozdziału zajmiemy się sposobami dostrajania i modyfikowania sieci w celu uzyskiwania zróżnicowanych profili osobowych agentów.

Projektowanie cech osobowości

Różnicowanie cech osobowych agentów sterowanych za pomocą sieci behawioralnych może być dokonywane na trzy sposoby: poprzez zmianę wartości parametrów globalnych, zmianę potencjałów celów i zmianę topologii samej sieci.

Manipulowanie parametrami globalnymi

Modyfikowanie wartości parametrów globalnych pozwala na kontrolowanie dwóch cech osobowości: rozwagi (parametr θ) oraz konsekwencji postępowania (parametr β).

Ustawienie wysokiego progu aktywacji θ prowadzi do bardziej rygorystycznego wyboru akcji — jak pamiętamy, do wykonywania wybierane są jedynie te moduły, których iloczyn $h(a, e) = a*e$ przekracza próg aktywacji każdego z zasobów niezbędnych do wykonania. Im mniejsza wartość wspomnianego iloczynu, tym dłużej (więcej cykli aktywacyjnych) dany moduł oczekiwał będzie na wykonanie — próg aktywacji każdego zasobu zmniejsza się liniowo z czasem, o czym wcześniej wspominaliśmy. Dla zewnętrznego obserwatora jawi się to jako rozważne działanie agenta: podejmuje on bowiem namysłu tylko te działania, które prowadzą do wyznaczonego celu w sposób oczywisty i najbardziej efektywny.

Duża wartość β prowadzi natomiast do zachowania konsekwentnego (uporczywego). Agent zmienia swe postępowanie jedynie w przypadku spektakularnych bądź trwałych zmian w zachodzącym środowisku. Załóżmy, że (na potrzeby gry *Unreal Tournament*) chcielibyśmy skonstruować dwa profile osobowości: weterana i nowicjusza (rekruta). Weteran (z założenia) działa spokojnie oraz racjonalnie i dąży do maksymalizacji swych osiągnięć w dłuższej perspektywie czasowej; cechuje się dużą dozą samokontroli i uporu, dąży do zabicia jak największej liczby przeciwników, lecz nigdy za cenę własnego życia. Nowicjusz, kierując się — co prawda — podobnymi motywacjami, działa jednak bardziej impulsywnie, często działa w sposób niezbyt odpowiedni do warunków otaczającego środowiska. Brak mu wytrwałości i opanowania, tak charakterystycznych dla weterana.

Zatem osobowość weterana preferuje cel długofalowy — zdobycie jak największej liczby punktów w długiej serii rozgrywek; ten właśnie cel osiągnąć można za pomocą sieci, której schemat widnieje na rysunku 3.5.3. Nawiązując do przedniego akapitu — mamy tu do czynienia z archetypem wojownika-weterana: konsekwentnego w zabijaniu, troszczącego się o własną kondycję (w chwilach wolnych od niebezpieczeństwa) i zdecydowanego walczyć nawet w przypadku kontuzji, bez wpadania w panikę.

Zastanówmy się jak — w kategoriach sieci behawioralnej — przetransformować ów rys osobowości na osobowość nowicjusza. Obniżając parametr bezwładności (β), sprawiamy, że agent staje się bardziej impulsywny, a przez to żywiej reagujący na zmiany zachodzące w zewnętrznym środowisku. Agent trafiony przez przeciwnika natychmiast rezygnuje z ofensywy (na pewien czas), starając się unikać dalszych trafień; ścigany przez przeciwnika, za wszelką cenę stara się ratować własną kondycję, niezwłocznie korzystając z dostępnego zestawu medycznego. Takie postępowanie może być niezbyt racjonalne: przecież ścigający agenta przeciwnik może być (i często jest) ciężko ranny i wyczerpany, dla wygranej agenta wystarczyłby więc krótki atak przy użyciu młota pneumatycznego. A uniki nie zawsze są skuteczne, potrafią chronić jedynie przed ekstremalnym niebezpieczeństwem.

By sprawić, żeby działanie nowicjusza stało się mało racjonalne, obniżyliśmy globalny próg aktywacji (θ). Powoduje to bardziej dynamiczne aktywowanie modułów, przez co jakość wyboru akcji obniża się zdecydowanie, sam zaś wybór staje się dość chaotyczny — moduły przekraczające próg aktywacji zasobów aktywowane są niezależnie od siebie, bez jakiejkolwiek koordynacji. W rezultacie możemy np. zaobserwować, że agent oddaje serię strzałów do przeciwnika, mimo iż ten ostatni znajduje się na tyle blisko, że jedno uderzenie młotem wystarczyłoby w zupełności. Przy ekstremalnie małej wartości $\theta = 0{,}1$ agent przejawia tendencję do oczekiwania w miejscu (`Stand`), zamiast penetrować zakamarki terenu (`Explore`).

Ostatecznie najlepszym zestawem parametrów, jaki udało nam się dobrać w celu symulowania zachowań nowicjusza, jest: $\gamma = 1{,}0$, $\delta = 0{,}9$, $\beta = 0{,}1$, $\theta = 0{,}25$ oraz $\Delta\theta = 0{,}1$.

Manipulowanie potencjałami celów

Załóżmy teraz, że projektant gry, zachęcony łatwością wykreowania dwóch opisanych osobowości, postanawia stworzyć dwie kolejne: samuraja i dzikusa.

Samuraj to człowiek zimny, uparty i agresywny. Śmierć w walce jest dla niego sprawą najwyższego honoru, podobnie jak przestrzeganie reguł „czystej" walki. Jego podstawowy cel to zabicie przeciwnika, nawet za cenę własnego życia. Podejmując walkę z jednym przeciwnikiem, walczy konsekwentnie i nie zważa na własne obrażenia i cierpienie, nie atakuje też innych przeciwników.

Dzikus także jest wysoce agresywny, brak mu jednak dyscypliny i wytrwałości właściwej samurajowi. Gdy tylko pojawi się na arenie gry, wściekle i bez opamiętania atakuje wszystkich widocznych przeciwników; niewrażliwy na ból, nie troszczy się o własne bezpieczeństwo i nie robi nic w celu utrzymania swej kondycji.

Samuraj to charakter pokrewny weteranowi — konsekwentny, doświadczony i rozważny; w celu dokonania transformacji między tymi dwiema osobowościami manipulować będziemy wartościami potencjału poszczególnych celów. Ustawiając potencjał celu `EnemyHurt` na 1,0, celu `Not IAmBeingShot` na 0,6, celu `Not LowHealth` na 0,5 i `HaveHighHealth` na 0,4 uzyskaliśmy agenta, który konsekwentnie posuwa się w kierunku przeciwnika. Ostrzeliwuje go i w końcu dobija młotem pneumatycznym (`FinalizeWithHammer`); nie stosuje uników i nie przerywa walki w celu opatrywania własnych ran. O własnym

bezpieczeństwie i kondycji samuraj myśli tylko wówczas, gdy w jego polu widzenia znajduje się żaden przeciwnik. Jego wola walki, silniejsza od cierpienia i poczucia zagrożenia, połączona jest z rozsądkiem: strzela do odległego przeciwnika, w walce taktowej sięga po młot pneumatyczny.

Samuraj posłużył nam też jako pierwowzór osobowości dzikusa — obaj mają przecież podobne cele do osiągnięcia. Obniżając jeszcze bardziej potencjał celu Not Low sprawiliśmy, że dzikus stał się skrajnie niewrażliwy na ból; by wyzwolić czającą się w jego wnętrzu wściekłość, obniżyliśmy zarówno bezwładność (β), jak i globalny próg aktywacji (θ). Niska bezwładność czyni agenta wysoce reaktywnym, niski próg aktywacji zasobów skutkuje (podobnie jak w przypadku nowicjusza) chaotycznymi, czasem bezsensownymi akcjami (jak strzelanie do przeciwnika stojącego dwa kroki dalej, choć do dyspozycji jest młot pneumatyczny). Uzyskaliśmy to, co chcieliśmy — oszalałą, zdeterminowaną walkę z użyciem każdej broni, bez względu na wynikające z niej zagrożenia.

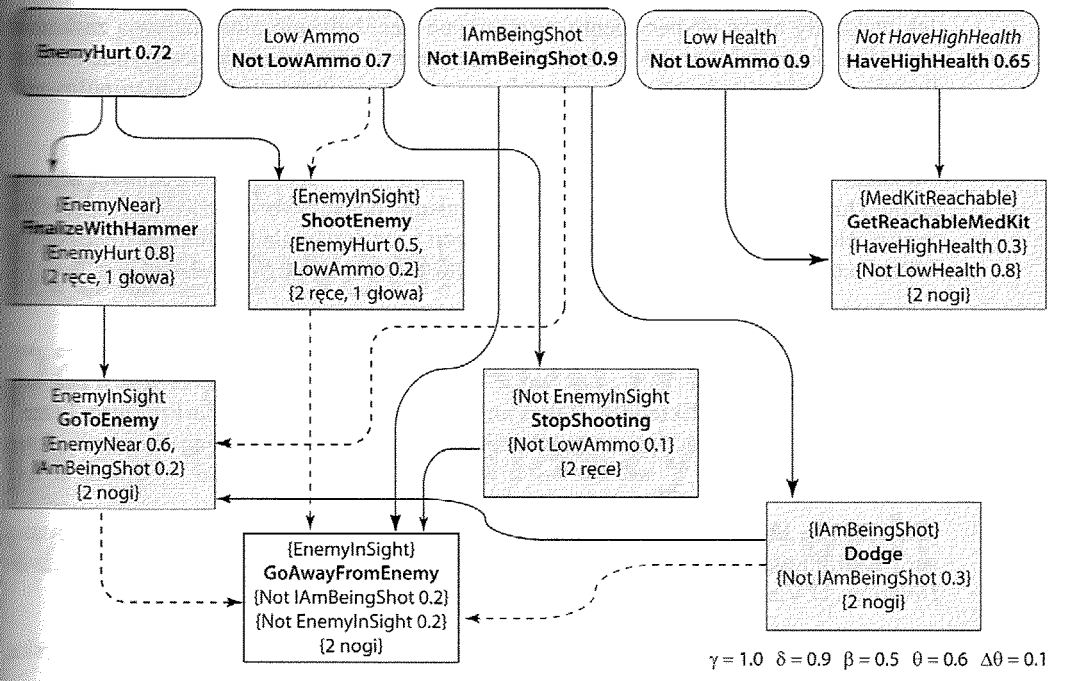
Dodawanie nowych modułów behawioralnych

Mamy już weterana, nowicjusza, samuraja i dzikusa, do utworzenia scenerii prawdziwej walki brakuje nam tylko jednego fundamentalnego archetypu — tchórza. Dążącego za wszelką cenę do ratowania własnej skóry, uciekającego przed niebezpieczeństwem, unikającego konfrontacji i najwyżej ceniącego sobie własną kondycję.

Jak z weterana zrobić tchórza? To proste: wystarczy obniżyć potencjał celu Enemy Hurt i podwyższyć potencjał celu HaveHighHealth. Parametry globalne pozostaną niezmienione, bowiem tchórz, podobnie jak weteran (i samuraj), jest w swym postępowaniu (a jakże) rozsądny i konsekwentny.

Strategię tchórza opisać można następująco. Rozpoczyna od penetracji terenu i, spostrzegłszy przeciwnika w polu widzenia, rozpoczyna ostrzeliwanie. Przeciwnik kontratakuje, w odpowiedzi na co tchórz próbuje uniknąć trafienia; gdy jednak zostanie trafiony, natychmiast sięga po dostępny zestaw medyczny. Jeżeli w jego zasięgu brakuje dostępnych zestawów medycznych, kontynuuje uniki i walczy aż do wyczerpania kondycji, po czym salwuje się ucieczką i poszukuje dostępnego zestawu medycznego, zdając sobie sprawę, iż ten znajdować się może nawet bardzo daleko.

Choć agent-tchórz ceni sobie głównie własne bezpieczeństwo i trudno doszukiwać się w jego filozofii przejawów dzielności, to jednak w naszej sieci brakuje kluczowego elementu jego specyfikacji: czynnego unikania zaangażowania w walkę. By zaimplementować ten aspekt jego osobowości, dodaliśmy do sieci nowy moduł GoAwayFromEnemy. Moduł ten wykonywany jest współbieżnie z modułem ShootEnemy, bowiem uciekający przed wrogiem tchórz osłania się za pomocą ostrzeliwania. Sieć behawioralną odzwierciedlającą zachowanie tchórza przedstawiliśmy schematycznie na rysunku 3.5.4; zauważmy, że dodanie do sieci nowego zachowania było operacją czysto modularną, niewymagającą żadnych dodatkowych zabiegów dostosowawczych — właściwe relacje między modułami zapewnione są automatycznie przez samą architekturę sieci.



Rysunek 3.5.4. *Sieć behawioralna odzwierciedlająca osobowość tchórza*

Konkluzja

Jak widzieliśmy, rozszerzona sieć behawioralna stanowi doskonały mechanizm selekcji akcji dla skomplikowanego agenta, kierującego się złożonymi celami i działającego w ciągłym, dynamicznym środowisku.

W przeprowadzonych eksperymentach zweryfikowaliśmy podstawowe właściwości tego mechanizmu selekcyjnego — konsekwencja w postępowaniu, wykorzystywanie okazji, preferencja akcji służących spełnianiu wielu celów, właściwe rozwiązywanie konfliktów, grupowanie akcji we właściwe sekwencje i prawidłowe uruchamianie akcji współbieżnych.

Z perspektywy agenta zaletą rozszerzonej sieci behawioralnej jest jej modularność i łatwe rozbudowywanie o nowe cele i zachowania. Projektant może skoncentrować się na pojedynczym module w danej chwili i w ten sposób sukcesywnie budować bibliotekę podstawowych zachowań. Interakcje między modułami tej biblioteki zapewniane są automatycznie przez architekturę sieci, co — z jednej strony — zwalnia projektanta z potrzeby przewidywania tych interakcji a priori, z drugiej zaś, prowadzić może do interesujących, często zaskakujących efektów w przebiegu gry.

Projektując agenta gry, rozpoczynamy od określenia celów odzwierciedlających jego cele i motywacje. Następnie implementujemy moduły, dla których cele te znajdują się na liście efektów. Ostatecznie, zmieniając wartości potencjału celów i wartości parametrów globalnych, dostrajamy poszczególne aspekty zachowania agenta, aż do osiągnięcia pożądanego profilu jego osobowości.

Jak wykazały nasze eksperymenty, rozszerzone sieci behawioralne stanowią znakomite narzędzie do projektowania zachowań stereotypowych: są proste koncepcyjnie i umożliwiają projektowanie zróżnicowanych agentów za pomocą jedynie manipulowania wartościami stałych.

Na zakończenie musimy zaznaczyć, że rozszerzone sieci behawioralne spełniają wymogi architektury planowania akcji sterowanych celami, dodatkowo umożliwiając specyfikowanie celów zależnych od konkretnych sytuacji i rozstrzyganie konfliktów między tymi celami, co predestynuje je do wielorakich zastosowań w dziedzinie robotyki.

Literatura zalecana

[Dorer99] K. Dorer „Extended Behavior Networks for the Magma Freiburg Team". RoboCup-99 Team Descriptions for the Simulation League, Linkoping University Press, 1999, str. 79 – 83.

[Dorer04] K. Dorer „Extended Behavior Networks for Behavior Selection in Dynamic and Continuous Domains". *Proceedings of the ECAI Workshop Agents in Dynamic Domains*. U.Visser i in., (Hrsg.) Valencia, Hiszpania, 2004.

[Goetz97] P. Goetz „Attractors in Recurrent Behavior Networks". Praca doktorska. University of New York, Buffalo, 1997.

[Maes89] P. Maes „How to do the Right Thing". *Connection Science Journal*, t.1, nr 3, 1989.

[Müller01] K. Müller „Roboterfußball: Multiagentensystem". Luty 2001, CS Freiburg, praca dyplomowa. Uniwersytet Freiburg, Niemcy.

[Nebel03] B. Nebel i Y. Babovich „Goal-Converging Behavior Networks and Self-Solving Planning Domains, or: How to Become a Successfull Soccer Player". s.l. *IJCAI03*, 2003.

[Pinto5-a] Hugo Pinto „An Extended Behavior Networks for a Game Agent". Anais do Quinto Encontro Nacional de Inteligência Artificial, Brazylia, 2005.

[Pinto5-b] H. Pinto i L. O. Alvares „Extended Behavior Networks and Agent Personality: Investigating the Design of Character Stereotypes in the Game Unreal Tournament". *Proceedings of the Fifth International Working Conference on Intelligent Virtual Agents*. Kos, Grecja, 2005.

[Rhodes96] Bradley Rhodes „PHISH-Nets; Planning Heuristically in Situated Hybrid Networks". Praca magisterska, Massachusetts Institute of Technology, Boston, 1996.

[Tyrrell93] Toby Tyrrell „Computational Mechanisms for Action Selection". Praca doktorska. University of Edinburgh, Wlk. Brytania, 1993.