PROJEKTOWANIE AI GRY

Jak dotąd widzieliśmy całą paletę technik sztucznej inteligencji i infrastrukturę, która umożliwia sztucznej inteligencji. Wspomniałem , że rozwój sztucznej inteligencji w grach jest mieszanką technik i infrastruktury ze sporą dawką rozwiązań ad hoc, heurystyki i fragmentów kodu, które wyglądają jak hacki. Tu przyjrzymy się, w jaki sposób wszystkie bity są stosowane w prawdziwych grach i jak stosowane są techniki, aby uzyskać rozgrywkę, której chcą programiści. W oparciu o gatunek po gatunkach przyjrzę się oczekiwaniom graczy i pułapkom związanym ze sztuczną inteligencją w grze. Nie uwzględniono tutaj żadnych technik, a jedynie ilustracją, w jaki sposób stosowane są techniki w innych miejscach książki. Klasyfikacja tutaj jest dość wysoka i luźna, a niektóre gry mogą być reklamowane jako inny gatunek. Ale z punktu widzenia AI, istnieje stosunkowo ograniczony zestaw rzeczy do osiągnięcia i odpowiednio pogrupowałem gatunki. Zanim zagłębimy się w każdy gatunek, warto przyjrzeć się ogólnemu procesowi projektowania sztucznej inteligencji w twojej grze.

PROJEKT

W tym tekście pracuję z tym samym modelem sztucznej inteligencji. Gdy projektowana jest sztuczna inteligencja dla tytułu, zespół zazwyczaj pracuje z docelowym zestawem zachowań opisanych w dokumencie projektowym. Programiści AI (często liderzy techniczni AI) opracowują najprostszy zestaw technologii, które wesprą wizję projektantów. To może się powtarzać, jeśli coś jest szczególnie trudne, lub istnieje prosta okazja do uzyskania bardziej wyrafinowanego wyniku. Gdy zespół przekona się, że rozumie wymagania, jakie nakładają te zachowania, wybierane są technologie do ich wdrożenia, wraz z podejściem do integracji tych technologii. Zaimplementowano strukturę, w tym warstwę integracyjną między planowaną sztuczną inteligencją a resztą silnika gry. Początkowo zachowania zastępcze mogą być używane dla postaci, ale przy istniejącej infrastrukturze różni programiści lub projektanci poziomów mogą zacząć pracować nad dopracowywaniem postaci. To jest oczywiście ideał, plan działania, gdybym miał wolną rękę nad projektem. W rzeczywistości napotkasz ograniczenia z wielu różnych kierunków, które wpłyną na Twój plan podejścia. W szczególności kamienie milowe wydawcy i zwiastun rozgrywki na konferencjach lub wydaniach z wczesnym dostępem oznaczają, że funkcjonalność i zachowania muszą zostać zaimplementowane na wczesnym etapie cyklu rozwoju. W wielu, jeśli niestety nie w większości projektów, takie treści są szybko wdrażane tylko w celu osiągnięcia kamienia milowego, a następnie usuwane i przepisywane później. W znacznej liczbie projektów, szybki i brudny kod zostaje tak załatany i zhakowany, że staje się niemożliwy do chirurgicznego usunięcia i staje się sztuczną inteligencją, która jest wysyłana. Tego rodzaju problemy są normalną częścią branży i zdarzają się każdemu. Czasu jest mało, zasoby są ograniczone i zawsze można zrobić więcej. Gdyby istniała tajna kabała ekspertów AI w grach, nie zostałbyś wciągnięty w czarną piłkę za wysyłanie od czasu do czasu zhakowanego i niedopracowanego kodu AI. Z drugiej strony jakość zostaje zauważona: możesz zrobić świetną karierę, jeśli myślisz z wyprzedzeniem i budujesz niezawodną i skuteczną sztuczną inteligencję.

PRZYKŁAD

W tej sekcji na przykładzie hipotetycznej gry omówię potrzeby projektowe w dwóch etapach (wymagane zachowania i technologie do ich osiągnięcia) hipotetycznej gry. Gra jest prosta z punktu widzenia rozgrywki, ale wymagania AI są zróżnicowane. Nasza gra nazywa się „Nawiedzony dom” i nic dziwnego, że jest osadzona w nawiedzonym domu. Jest to dobrze znany nawiedzony dom, a ludzie z daleka płacą pieniądze, aby go odwiedzić. Gracz jest właścicielem domu, a jego zadaniem jest utrzymanie klientów płacących poprzez zarządzanie strachami w domu, upewniając się, że odwiedzający dostaną straszydła, których szukają. Goście przybywają do domu, a celem gracza jest zmuszenie ich do ucieczki w panice. Aby to zrobić, gracz otrzymuje wybór zjaw i mechanicznych

sztuczek do zastosowania w domu. Poprzedni odwiedzający nieuchronnie dzielą się swoimi doświadczeniami, a inni przyjdą, aby obalić lub naśladować ich przerażenie. Gracz musi również starać się, aby goście nie natknęli się na tajemnice domu, sztuczki rzemieślnicze, lustra weneckie, maszyny do dymu i pokój wspólny duchów. Wariant tego pomysłu można zobaczyć w Ghost Master gdzie prezentowane są różne domy z różnymi mieszkańcami. Mieszkańcy nie spodziewają się, że będą się bać i będą podążać własnym życiem podobnym do Simów. Ma też podobieństwa do gier takich jak Dungeon Keeper i Evil Genius .

OCENA ZACHOWAŃ

Pierwszym zadaniem jest zaprojektowanie zachowań, które będą wyświetlać postacie w twojej grze. Jeśli pracujesz nad własną grą, prawdopodobnie jest to część Twojej wizji projektu. Jeśli pracujesz w studiu deweloperskim, prawdopodobnie będzie to praca projektanta gry. Podczas gdy projektant gry będzie miał pewne pomysły na to, jak powinny zachowywać się postacie w grze, z mojego doświadczenia wynika, że rzadko są one osadzone w kamieniu. Często projektanci nie rozumieją tego, co wydaje się trywialne, ale jest naprawdę trudne (i dlatego powinno być uwzględnione tylko wtedy, gdy jest to centralny punkt gry) oraz wiele pozornie trudnych, ale prostych dodatków, które można wprowadzić, aby poprawić zachowanie postaci. Zachowanie postaci w grze będzie naturalnie ewoluować w miarę wdrażania i próbowania nowych rzeczy. Dotyczy to nie tylko projektów hobbystycznych lub gier z długimi fazami badań i rozwoju; dotyczy to również projektu rozwojowego ze stałymi pomysłami i napiętą skalą czasową. Mając najlepszą wolę na świecie, nie zrozumiesz w pełni potrzeb AI w grze, zanim zaczniesz ją rozwijać. Warto od początku zaplanować pewien stopień elastyczności. Na przykład tworzenie sztucznej inteligencji ze stałym zestawem danych wejściowych z gry to po prostu proszenie o późne noce pod koniec projektu (i tak się wydarzą, więc po co o nie prosić?). Nieuchronnie projektanci będą potrzebować dodatkowych danych wejściowych AI w niewygodnym czasie, a kod AI będzie wymagał przeróbki. Z tego powodu zawsze lepiej jest błądzić po stronie elastyczności niż prędkość w początkowym planie wdrożenia. Znacznie łatwiej jest później zoptymalizować niż de-optymalizować splątany kod, aby zachować elastyczność w ostatniej chwili. Tak więc, zaczynając od zestawu zachowań, które chcemy zobaczyć, musimy odpowiedzieć na kilka pytań dla każdego z elementów modelu AI:

• Ruch

– Czy nasze postacie będą reprezentowane indywidualnie (jak w większości gier), czy też zobaczymy tylko ich efekty grupowe (jak na przykład w grach symulacyjnych miast)?

– Czy nasze postacie będą musiały poruszać się po swoim otoczeniu w mniej lub bardziej realistyczny sposób? Czy możemy po prostu umieścić je tam, gdzie chcemy, aby trafiły (na przykład w grze turowej opartej na kafelkach)?

– Czy ruch postaci będzie musiał być symulowany fizycznie, jak na przykład w grze samochodowej? Jak realistyczna musi być fizyka (pamiętając, że zazwyczaj znacznie trudniej jest zbudować algorytmy ruchu do pracy z realistyczną fizyką, niż poprawić fizykę, aby była mniej realistyczna dla postaci AI)?

– Czy postacie będą musiały ustalić, dokąd się udać? Czy uda im się po prostu wędrować, podążać ścieżkami wyznaczonymi przez projektantów, przebywać tylko na jednym małym obszarze lub ścigać inne postacie?

A może potrzebujemy postaci, aby móc zaplanować swoją trasę przez cały poziom za pomocą systemu odnajdywania ścieżki?

– Czy na ruch postaci będą musiały wpływać inne postacie? Czy pogoń/unikanie wystarczą, aby sobie z tym poradzić, czy też postacie muszą koordynować się lub poruszać w formacjach?

• Podejmowanie decyzji

– Jest to zazwyczaj obszar, w którym projektanci AI dają się ponieść najbardziej. Kuszące jest, aby w fazie przedprodukcyjnej gry chcieć wykorzystać wszelkiego rodzaju egzotyczne nowe techniki. Najczęściej ostateczna gra jest dostarczana z automatami stanów lub zakodowanymi na stałe skryptami, które uruchamiają wszystkie ważne rzeczy.

– Jaki jest pełen zakres różnych akcji, które twoje postacie mogą wykonywać w grze?

– Ile odrębnych stanów będzie mieć każda z twoich postaci? Innymi słowy, w jaki sposób te działania są zgrupowane, aby spełnić cele postaci? Zauważ, że nie zakładam, że użyjesz tutaj ani maszyn stanowych, ani zachowań zorientowanych na cel. Niezależnie od tego, co kieruje twoimi postaciami, powinny wydawać się, że mają cele, a gdy działają, aby osiągnąć jeden cel, można je traktować jako będące w jednym stanie.

– Kiedy twoja postać zmieni swoje zachowanie, przejdzie do innego stanu lub wybierze inny cel do naśladowania? Co spowoduje te zmiany? Co musi wiedzieć, aby zmienić się we właściwym czasie?

– Czy bohaterowie będą musieli patrzeć w przyszłość, aby wybrać najlepszą decyzję? Czy będą musieli planować swoje działania lub prowadzić działania, które tylko pośrednio prowadzą do ich celów? Czy te działania wymagają planowania działań, czy może objąć je bardziej złożone podejście oparte na stanach lub regułach?

– Czy twoja postać będzie musiała zmienić decyzje, które podejmuje w zależności od tego, jak zachowuje się gracz? Czy będzie musiał reagować w oparciu o pamięć działań gracza, wykorzystując jakiś rodzaj nauki?

• Taktyczna i strategiczna sztuczna inteligencja

- Czy twoje postacie muszą rozumieć właściwości poziomu gry na dużą skalę, aby podejmować rozsądne decyzje? Czy musisz reprezentować im sytuacje taktyczne lub strategiczne w sposób, który pozwoli im wybrać odpowiednie zachowanie?

- Czy twoi bohaterowie muszą ze sobą współpracować? Czy muszą wykonywać czynności we właściwych sekwencjach, w zależności od ich czasu?

- Czy twoi bohaterowie potrafią myśleć samodzielnie i nadal przejawiać zachowanie grupowe, którego szukasz? A może potrzebujesz decyzji, które należy podejmować dla grupy postaci na raz?

Przykład

W Nawiedzonym Domu proponuję następujący wstępny zestaw odpowiedzi na te pytania:

• Ruch

– Postacie będą reprezentowane indywidualnie, poruszając się autonomicznie po swoim otoczeniu. Nie potrzebujemy realistycznej symulacji fizycznej. Możemy poradzić sobie z algorytmami ruchu kinematycznego, a nie pełnymi zachowaniami sterującymi. Postacie często będą chciały udać się do określonej lokalizacji (na przykład wyjścia), która może wymagać nawigacji po domu, więc będziemy potrzebować odnajdywania ścieżki.

• Podejmowanie decyzji

– Postacie mają niewielki zakres możliwych akcji. Mogą skradać się, biegać lub stać nieruchomo (skamieniałe). Mogą badać przedmioty lub „działać na nich”: każdy przedmiot ma maksymalnie jedną akcję, którą można na nim wykonać (można na przykład przełączyć włącznik światła lub otworzyć drzwi). Mogą również pocieszać inne osoby w domu.

– Postacie będą miały cztery ogólne typy zachowań: przestraszone zachowanie, w którym będą próbować odzyskać rozum; ciekawe zachowanie, w którym będą badać przedmioty i eksplorować; zachowania społeczne, w których będą próbowali utrzymać grupę razem i pocieszać zainteresowanych członków; i znudzone zachowanie, gdzie kierują się do biura obsługi klienta i poprosić o zwrot pieniędzy.

– Postacie zmienią swoje zachowanie w zależności od poziomu strachu. Każda postać ma poziom strachu. Gdy postać przekroczy próg, zacznie się przerażać. Kiedy postać znajduje się w pobliżu innej przestraszonej postaci, wejdzie w zachowanie społeczne. Jeśli poziom strachu postaci spadnie bardzo nisko, to się nudzi. W przeciwnym razie będzie w trybie ciekawskim.

– Postacie zmienią swój poziom strachu, widząc, słysząc lub wąchając dziwne rzeczy. Każdy straszak i sztuczka mają intensywność dziwności w każdym z tych trzech zmysłów. Postacie muszą być informowane, kiedy coś widzą, słyszą lub powąchają i jak dziwnie się to wydaje.

– Bohaterowie będą starali się odkrywać miejsca, w których wcześniej nie byli, lub wrócą do miejsc, które wcześniej lubili. Powinni śledzić odwiedzane miejsca i ciekawe miejsca. Ciekawe miejsca można podzielić między wiele grup, aby przedstawić plotki o dobrych strachach.

• Taktyczna i strategiczna sztuczna inteligencja

– Postacie muszą unikać miejsc, o których wiedzą, że są przerażające, gdy próbują odzyskać rozum. Podobnie będą unikać nudnych obszarów, gdy szukają działania.

WYBÓR TECHNIKI

Dzięki odpowiedziom na pytania behawioralne będziesz miał dobre pojęcie o tym, jak daleko musisz się posunąć w sztucznej inteligencji. Być może wiesz, na przykład, czy potrzebujesz odnajdywania ścieżki i jakiego rodzaju zachowania ruchowe są wymagane, ale niekoniecznie jakiego algorytmu odnajdywania ścieżki lub którego systemu arbitrażu sterującego użyć. To kolejny etap: zbudowanie kandydującego zestawu technologii, z których zamierzasz korzystać. Większość z nich jest dość prosta. Jeśli zdecydowałeś, że potrzebujesz pathfindingu, to A* jest oczywistym wyborem. Jeśli wiesz, że postacie muszą poruszać się w szyku, potrzebujesz systemu ruchu szyku. Podejścia do podejmowania decyzji są nieco bardziej skomplikowane, ponieważ często istnieje kilka sposobów na uzyskanie tego samego efektu. Wybrane podejście może mieć więcej wspólnego z tym, jakie masz doświadczenie w tworzeniu, jakie narzędzia masz już licencjonowane lub jaki istniejący kod możesz ponownie wykorzystać. Jak widzieliśmy w rozdziale 5, nie ma twardych i szybkich zasad wyboru systemu podejmowania decyzji. Większość rzeczy, które możesz zrobić za pomocą jednego systemu, możesz zrobić z innymi. Jeśli budujesz swoją sztuczną inteligencję od zera, sugeruję zacząć od prostej techniki, takiej jak drzewa zachowań lub maszyny stanów lub od prostej kombinacji tych dwóch, chyba że wiesz o konkretnej rzeczy, którą chcesz zrobić, a której nie można osiągnąć ich. Ich elastyczność udowodniła swoją wartość tak wiele razy, że osobiście muszę mieć dobry powód, aby użyć czegoś bardziej złożonego. Ostatnio tym „dobrym powodem” była najczęściej dostępność wspieranych narzędziowo drzew zachowań w używanym przeze mnie silniku. Na tym etapie staraj się nie dać się wciągnąć w ponowne zaprojektowanie zidentyfikowanych zachowań. Kusząca jest myśl, że gdybyśmy zastosowali taką a taką egzotyczną technikę, to moglibyśmy pokazać takie a takie fajne zachowanie. Ważne jest,

aby połączyć obietnicę fajnych efektów z możliwością solidnego działania pozostałych 95% sztucznej inteligencji.

Przykład

W Nawiedzonym Domu możemy spełnić wymagania naszych zachowań za pomocą następującego zestawu technologii:

Ruch

– Postacie będą się poruszać za pomocą kinematycznych algorytmów ruchu. Mogą wybrać dowolny kierunek poruszania się z jedną z dwóch prędkości ruchu.

– W trybach ciekawskich i przestraszonych wybierają swój cel ruchu jako pomieszczenie i używają A*, aby znaleźć tam drogę. Będą używać ścieżki podążającej za zachowaniem, aby podążać trasą. Użyjemy wykresu punktów orientacyjnych, aby dopasować się do taktycznej i strategicznej sztucznej inteligencji poniżej.

– W trybie społecznościowym będą kierować się do przerażonych postaci, które widzą, używając kinematycznego zachowania poszukiwania.

• Podejmowanie decyzji

– Postacie użyją bardzo prostej maszyny skończonych stanów, aby określić swój szeroki wzorzec zachowania, a w każdym stanie drzewo zachowań, aby określić, co właściwie z tym zrobić.

– Maszyna państwowa ma cztery stany: przestraszony, ciekawy, towarzyski i znudzony. Przejścia są oparte wyłącznie na poziomie strachu postaci i innych postaci w linii wzroku.

– W każdym trybie może być dostępnych wiele akcji. W trybie ciekawskim postać może badać lokacje lub obiekty; w trybie strachu chcą wybrać najlepszy sposób na znalezienie bezpiecznego miejsca do zebrania myśli. Każde z tych zachowań jest zaimplementowane jako drzewo decyzyjne z różnymi strategiami wybranymi przez selektory. Każda strategia może z kolei zawierać wiele elementów, które można dodać do węzłów sekwencji w drzewie.

• Taktyczna i strategiczna sztuczna inteligencja

– Aby ułatwić naukę przerażających i bezpiecznych lokalizacji, prowadzimy mapę z punktami na poziomie. Kiedy postacie zmieniają swój przestraszony stan, zapisują wydarzenie na mapie. Jest to dokładnie ten sam proces, co tworzenie mapy fragmentów z część u 6.

• Światowy interfejs

– Postacie muszą uzyskać informacje o widokach, zapachach i dźwiękach dziwnych wydarzeń w grze. Powinna to być symulacja zarządzania zmysłami (kierownik ds. zmysłów regionu byłby w porządku).

– Postacie potrzebują również informacji o dostępnych akcjach, które mogą wykonać, gdy są w trybie ciekawości. Postać może poprosić o listę obiektów, z którymi może wchodzić w interakcje, a my możemy dostarczyć te informacje z bazy danych obiektów w grze. Nie musimy symulować postaci, która widzi i rozpoznaje te obiekty.

• Zarządzanie wykonaniem

– Istnieją dwie technologie, odnajdywanie ścieżki i zarządzanie zmysłami. Oba są czasochłonne.

– Mając tylko kilka pokoi w domu, odnalezienie drogi przez jednostkę nie potrwa długo. Jednak w domu może być wiele postaci, więc możemy skorzystać z puli kilku planistów (można to zrobić) i kolejkować żądania odszukania ścieżki. Kiedy postać prosi o ścieżkę, czeka, aż planista jest wolny, a następnie otrzymuje swoją ścieżkę za jednym zamachem. Nie potrzebujemy żadnych algorytmów do wyszukiwania ścieżek.

– System zarządzania zmysłami jest wywoływany przy każdej klatce i stopniowo aktualizowany. Jest to z założenia algorytm w dowolnym momencie rozłożony na wiele ramek.

– W domu może być jednocześnie wiele postaci (powiedzmy dziesiątki). Każda postać działa stosunkowo wolno; nie musi przetwarzać całej swojej sztucznej inteligencji w każdej klatce. Możemy uniknąć korzystania ze złożonego hierarchicznego systemu planowania i po prostu zaktualizować kilka różnych znaków w każdej klatce. Przy 5 zaktualizowanych znakach na klatkę, 50 znakach w grze i renderowaniu 30 klatek na sekundę, postać będzie musiała czekać mniej niż pół sekundy między aktualizacjami. To opóźnienie może być rzeczywiście przydatne; gdy postacie czekają ułamki sekundy, zanim zareagują na strach, symuluje ich czas reakcji.

Skończyło się na tym, że mam tylko kilka modułów, które wymagają implementacji w tej grze. System zarządzania zmysłami jest prawdopodobnie najbardziej złożony, pozostałe są bardzo standardowe i składają się z prostych elementów. Udało mi się nawet włączyć generator liczb losowych: pierwszą technikę sztucznej inteligencji, którą poznaliśmy w rozdziale 2.

ZAKRES JEDNEJ GRY

Biorąc pod uwagę zakres technologii, można by się spodziewać, że uczynię Nawiedzony Dom bardziej złożonym, opierając się na sprytnym wykorzystaniu wielu różnych algorytmów. Ostatecznie jedyną nieco egzotyczną rzeczą w naszym projekcie jest system zarządzania zmysłami służący do powiadamiania postaci o dziwnych zdarzeniach. W rzeczywistości sztuczna inteligencja w grach działa w ten sposób. Większość pracy zajmują dość proste techniki. Jeśli szukasz konkretnych efektów rozgrywki opartych na sztucznej inteligencji, możesz zastosować jedną lub dwie nietypowe techniki. Jeśli projektujesz grę z sieciami neuronowymi, zarządzaniem zmysłami, potokami sterującymi i systemem eksperckim opartym na Rete, prawdopodobnie nadszedł czas, aby skupić się na tym, co jest naprawdę ważne w twojej grze. Każda z bardziej niezwykłych technik opisanych w tej książce jest kluczowa w niektórych grach i może stanowić różnicę między nudną grą a naprawdę zgrabnym zachowaniem postaci. Ale, podobnie jak dobra przyprawa, jeśli nie są używane oszczędnie, aby dodać smaku, mogą zepsuć produkt końcowy. W pozostałej części tego część u przyjrzę się gamie gier komercyjnych w różnych gatunkach. W każdym przypadku postaram się skupić na technikach, które czynią ten gatunek niezwykłym: gdzie nowe innowacje mogą naprawdę coś zmienić. W tym rozdziale omówiono tylko najważniejsze z komercyjnego punktu widzenia gatunki gier, chleb powszedni dla większości twórców sztucznej inteligencji. Ostatni część książki, część 15, przedstawia inne gatunki gier, w których sztuczna inteligencja ma za zadanie zapewnić rozgrywkę. Nie są to duże gatunki z tysiącami tytułów i nie są najlepiej sprzedające się, ale są interesujące dla programisty AI, ponieważ rozciągają sztuczną inteligencję w sposób, w jaki nie mają zwykłych gatunków.

STRZELANKI

Strzelanki jedno- i trzyoosobowe to gatunki odnoszące największe sukcesy finansowe i były w takiej czy innej formie od czasu powstania pierwszych gier wideo. Są dobrym punktem wyjścia do omówienia sztucznej inteligencji używanej do postaci wroga w wielu innych gatunkach. Na końcu tej sekcji będę

opierał się na omówieniu strzelanek AI, aby poszerzyć nacisk na gatunki takie jak gry przygodowe, platformówki, walki wręcz i MMOG. Na początek będę się trzymać klasyczne strzelanki. Wraz z pojawieniem się Wolfenstein 3D i Dooma gatunek strzelanek stał się synonimem postaci poruszających się pieszo (prawdopodobnie z plecakami odrzutowymi, jak w Tribes II z kamery powiązanej z postacią gracza. Wrogowie zazwyczaj składają się ze stosunkowo niewielkiej liczby postaci na ekranie. Gry, które mają być przede wszystkim gracz kontra gracz (PvP), zoptymalizowane pod kątem meczów między ludźmi, mogą mieć wyrafinowaną sztuczną inteligencję, znaną jako „boty". Aby zapewnić uczciwość, te kontrolowane przez komputer postacie będą miały możliwości jak najbardziej podobne do tych z gracz. Strzelanki zapewniające tryb kampanii lub inne wyzwania typu gracz kontra środowisko (PvE) zwykle polegają na większej liczbie mniej złożonych wrogów. Najważniejsze potrzeby AI dla tego gatunku to:

1. Ruch - kontrola nad wrogami

2. Strzelanie - dokładna kontrola ognia

3. Podejmowanie decyzji - zazwyczaj proste maszyny stanowe

4. Percepcja - ustalanie, do kogo strzelać i gdzie są

5. Odnajdywanie ścieżek - często (ale nie zawsze) używane, aby umożliwić postaciom zaplanowanie ich trasy przez poziom

6. Taktyczna sztuczna inteligencja - często używana, aby umożliwić postaciom określenie bezpiecznych pozycji do poruszania się lub punktów osłony, z których można prowadzić ogień

Spośród nich dwa pierwsze są kluczowymi problemami występującymi we wszystkich grach tego gatunku. Możliwe jest stworzenie strzelanki przy użyciu tylko tych narzędzi, zwłaszcza strzelanek 2D, które wróciły do mody wraz z odrodzeniem gier niezależnych. Ale w 3D może to wyglądać na beznadziejnie naiwne. Przez ostatnie 15 lat gracze coraz częściej spodziewali się wrogów z pewnym wyrafinowaniem taktycznym (takim jak użycie osłony) i którzy używają odnajdywania ścieżki, aby uniknąć utknięcia na poziomie.

RUCH I STRZELANIE

Ruch jest najbardziej widoczną częścią zachowania postaci we wszystkich gatunkach gier. W strzelankach 3D, w których postacie są zazwyczaj duże na ekranie, ruch i animacja razem sygnalizują graczowi większość informacji o tym, co robi sztuczna inteligencja. Niektóre strzelanki bardzo mocno polegają na animacji, z niektórymi najbardziej złożonymi zestawami animacji. Może to obejmować różne rodzaje ruchu (pełzanie, czołganie się, sprint), przeładowywanie, gesty do sojuszników, a nawet aktorstwo głosowe. Nie jest niczym niezwykłym, że postacie łączą dziesiątki lub setki sekwencji animacji wraz z innymi kontrolerami, takimi jak odwrócona kinematyka lub fizyka szmacianych lalek. Postać w F.E.A.R. 2: Project Origin może działać, strzelać i patrzeć na wszystko w tym samym czasie. Pierwsze dwa to kanały animacji, a trzeci to animacja proceduralna kontrolowana przez kierunek, w którym patrzy postać (podobnie jak kierunek, ale nie ogólny ruch ramienia strzelającego). W No One Lives Forever 2 postacie ninja mają wyrafinowane zdolności ruchowe, które dodatkowo utrudniają synchronizację ruchu i animacji. Mogą wykonywać koła wozu, przeskakiwać nad przeszkodami i przeskakiwać między budynkami.

Proste poruszanie się po poziomie staje się wyzwaniem. Sztuczna inteligencja nie tylko musi wypracować trasę, ale także musi być w stanie podzielić ten ruch na animacje. Większość gier oddziela te dwie części: sztuczna inteligencja decyduje, gdzie się przenieść, a kolejny kawałek kodu zamienia to w animacje. Pozwala to sztucznej inteligencji na pełną swobodę ruchu, ale ma tę wadę, że pozwala na

występowanie dziwnych kombinacji animacji i ruchu, co może wydawać się drażniące dla gracza. Ta trudność została do tej pory rozwiązana przez dodanie bogatszej palety animacji, co zwiększa prawdopodobieństwo znalezienia rozsądnej kombinacji. Kilka gier, które używają języków skryptowych do kontrolowania swoich postaci, udostępnia sztucznej inteligencji te same elementy sterujące, z których korzysta gracz. Zamiast wyprowadzać pożądany ruch lub lokalizacje docelowe, sztuczna inteligencja musi określić, jak szybko porusza się do przodu lub do tyłu, obraca się, zmienia broń i tak dalej. Dzięki temu podczas tworzenia bardzo łatwo jest usunąć postać AI i zastąpić ją człowiekiem (na przykład grając przez sieć). Większość tytułów, w tym te oparte na licencjonowaniu najsłynniejszych silników gier, ma makropolecenia - na przykład:

1 sleep 3

2 gotoactor PathNodeLoc1

3 gotoactor PathNodeLoc2

4 agentcall Event_U_Wave 1

5 sleep 2

6 gotoactor PathNodeLoc3

7 gotoactor PathNodeLoc0

Ze względu na ograniczony, wewnętrzny charakter poziomów w wielu strzelankach, postacie prawie na pewno potrzebują jakiegoś sposobu znalezienia trasy. Może to być tak proste, jak instrukcje gotoactor w powyższym skrypcie Unreal lub może to być pełny system wyszukiwania ścieżek. Bez względu na to, jaką formę przybierze (później wrócimy do rozważań dotyczących odnajdywania ścieżek), należy podążać trasami. Przy dość skomplikowanej trasie postać może po prostu podążać ścieżką. Niestety poziom gry prawdopodobnie będzie dynamiczny. Postać powinna odpowiednio reagować na inne poruszające się postacie. Najczęściej robi się to za pomocą prostej siły odpychania między wszystkimi postaciami. Jeśli postacie podejdą zbyt blisko, rozsuną się. W Mace Griffin: Bounty Hunter [197] ta sama technika jest używana do uniknięcia kolizji między postaciami na ziemi i między bojowymi statkami kosmicznymi podczas sekcji głębokiej przestrzeni gry. W pomieszczeniach do tworzenia tras wykorzystywane jest odnajdywanie ścieżek. W kosmosie zamiast tego używany jest system ruchu formacji. Powódź w Halo i jej kontynuacje, a także kosmici w Obcy vs. Predator1 [167] poruszają się zarówno po ścianach i suficie, jak i po podłodze. Żaden z nich nie używa reprezentacji ściśle 2½-wymiarowej (2½D) do poruszania się postaci. Strzelanie AI jest kluczowe w strzelankach (nic dziwnego). Pierwsze dwa wcielenia Dooma były mocno krytykowane za niewiarygodnie celne strzelanie (twórcy spowolnili nadlatujące pociski, aby gracz mógł zejść z drogi; w przeciwnym razie celność byłaby przytłaczająca). Bardziej realistyczne gry, takie jak gry ARMA [89] i seria Far Cry , wykorzystują modele strzelania, które pozwalają postaciom chybić, czasem zmodyfikowaną, aby spudłować w ekscytujący sposób (tj. próbują nie trafić tam, gdzie gracz może zobaczyć pocisk).

PODEJMOWANIE DECYZJI

Podejmowanie decyzji jest powszechnie osiągane przy użyciu prostych technik, takich jak maszyny skończone lub drzewa zachowań. Mogą one być bardzo podstawowe w przypadku zachowań „widzianego gracza” i „niewidzianego gracza”.

Bardzo powszechnym podejściem do podejmowania decyzji w strzelankach jest opracowanie systemu skryptów dla botów. Wykonywany jest skrypt napisany w języku skryptowym specyficznym dla gry lub dostarczonym przez silnik. Scenariusz posiada cały szereg funkcji, dzięki którym może określić, co

postać może postrzegać. Są one zwykle implementowane przez bezpośrednie odpytywanie bieżącego stanu gry. Skrypt może następnie zażądać wykonania akcji, w tym odtwarzania animacji, ruchu, aw niektórych przypadkach żądań odszukania ścieżki. Ten język skryptowy jest następnie udostępniany użytkownikom gry w celu modyfikowania sztucznej inteligencji lub tworzenia własnych autonomicznych postaci. Jest to podejście zastosowane w Unreal i kolejnych grach, które zostało przyjęte w innych strzelankach (pierwszym przykładem, jaki zauważyłem, była gra fabularna Neverwinter Nights). W przypadku Sniper Elite Rebellion chciał zobaczyć wyłaniające się zachowanie, które było inne na każdą grę. Aby to osiągnąć, zastosowali szereg maszyn stanowych, operujących na punktach orientacyjnych na poziomie gry. Wiele z tych zachowań zależało od działań innych postaci lub zmieniającej się sytuacji taktycznej w pobliskich punktach trasy. Niewielka losowość w procesie podejmowania decyzji pozwoliła postaciom zachowywać się za każdym razem inaczej i działać w pozornej współpracy, bez konieczności korzystania z SI opartej na drużynie. Nieco inne podejście do autonomicznej sztucznej inteligencji powstało w No One Lives Forever 2. Mieszane maszyny stanów Monolith z zachowaniem zorientowanym na cel. Każda postać miałaby z góry określony zestaw celów, które mogłyby wpłynąć na jej zachowanie. Bohaterowie okresowo oceniali swoje cele i wybierali ten, który w tamtym czasie był dla nich najbardziej odpowiedni. Ten cel przejąłby wtedy kontrolę nad zachowaniem postaci. Wewnątrz każdego celu znajdowała się skończona maszyna stanu, która służyła do kontrolowania postaci, dopóki nie został wybrany inny cel. Gra wykorzystuje punkty orientacyjne (które nazywają węzłami), aby upewnić się, że postacie są we właściwej pozycji do zachowań, takich jak przeglądanie szafek na akta, używanie komputerów i włączanie świateł. Obecność tych punktów orientacyjnych w pobliżu postaci pozwala jej zrozumieć, jakie akcje są dostępne. Silnik AI Monolith był nadal rozwijany, dopóki firma nie została przejęta. W strachu. [144] zastosowano to samo zachowanie zorientowane na cel, ale gotowe maszyny stanowe zostały zastąpione przez silnik planowania, który stara się łączyć dostępne działania w taki sposób, aby osiągnąć cel. STRACH. posiadał jeden z pierwszych w pełni zorientowanych na cel systemów planowania działań. W Halo 2 [92] i późniejszych grach z tej serii drzewa decyzyjne były używane, aby umożliwić postaciom AI wykonywanie szczątkowego planowania podczas działania. Gdy węzły w selektorze w drzewie zachowań zawodzą, sztuczna inteligencja wraca do innych węzłów reprezentujących różne plany, dając AI taktyczne możliwości, które byłyby trudne do określenia za pomocą maszyny stanów.

POSTRZEGANIE

Percepcja jest czasami sfałszowana, umieszczając promień wokół każdej wrogiej postaci i sprawiając, że wróg „ożywa", gdy gracz jest w nim. To jest podejście przyjęte przez oryginalny Doom. Jednak po sukcesie Goldeneye 007 oczekiwano bardziej wyrafinowanej symulacji percepcji. Niekoniecznie oznacza to system zarządzania zmysłami, ale przynajmniej postacie powinny być informowane o tym, co się wokół nich dzieje, poprzez jakiś rodzaj wiadomości. W grach Tom Clancy's Ghost Recon symulacja percepcji jest znacznie bardziej złożona. System zarządzania zmysłami, który dostarcza informacji postaciom AI, bierze pod uwagę ilość zniszczonej osłony zapewnianej przez krzaki i testuje tło postaci, aby określić, czy ich kamuflaż pasuje. Osiąga się to poprzez zachowanie zestawu identyfikatorów wzorów dla każdego materiału w grze. Kontrola pola widzenia przechodzi przez dowolny częściowo przezroczysty obiekt, aż dotrze do testowanej postaci. Następnie wychodzi poza postać i określa następną rzecz, z którą się zderzy. Identyfikator kamuflażu i identyfikator materiału tła są następnie sprawdzane pod kątem zgodności. Gry Splinter Cell wykorzystują inną taktykę. Ponieważ jest tylko jedna postać gracza (w Ghost Recon jest ich wiele), każda sztuczna inteligencja po prostu sprawdza, czy jest widoczna. Każdy poziom może zawierać dynamiczne cienie, mgłę i inne efekty ukrywania. Postać gracza jest sprawdzana pod kątem każdego z nich, aby określić poziom ukrycia. Jeśli jest to poniżej pewnego progu, wroga sztuczna inteligencja zauważyła postać gracza. Poziom ukrycia nie uwzględnia tła, tak jak robią to gry Ghost Recon; jeśli postać stoi w ciemnym cieniu na środku jasnego

korytarza, to nie będzie widoczna, mimo że strażnikom będzie wyglądała jak wielka czarna postać na jasnym tle. Poziomy zostały zaprojektowane tak, aby zminimalizować liczbę przypadków, w których to ograniczenie jest oczywiste. Postacie AI w Splinter Cell również używają stożka wzroku do kontroli wzroku, a istnieje prosty model dźwięku, w którym dźwięk przemieszcza się w bieżącym pomieszczeniu do określonego promienia w zależności od głośności dźwięku. Bardzo podobne techniki są stosowane w serii gier Metal Gear Solid.

SZKOLENIE I TAKTYCZNA AI

W Soldier of Fortune 2: Double Helix linki w grafie odnajdywania ścieżek zostały oznaczone rodzajem akcji potrzebnej do ich przemierzenia. Gdy postać dotrze do odpowiedniego łącza na ścieżce, może zmienić zachowanie, tak aby wyglądało na to, że zna teren. Łącze może reprezentować przeszkodę do przeskoczenia, drzwi do otwarcia, barierę do przebicia lub ścianę do zjazdu. Odpowiedzialny zespół AI, Christopher Reed i Ben Geisler, nazywa to podejście „nawigacją osadzoną". Wprowadzanie do strzelanek pewnego rodzaju taktyk punktowych staje się niemal powszechne. W oryginalnym Half-Life sztuczna inteligencja używa punktów orientacyjnych, aby ustalić, jak otoczyć gracza. Grupa postaci AI zostanie skoordynowana tak, aby zajmowały zestaw dobrych pozycji obronnych, które otaczają obecną lokalizację gracza, jeśli to możliwe. W grze postać AI często wykonuje desperacki bieg obok gracza, aby zająć pozycję oskrzydlającą. O ile wrogie postacie nie zawsze pędzą z graczem, jak w oryginalnym Doomie, prawdopodobnie będziesz musiał zaimplementować warstwę odnajdywania ścieżek. Poziomy wewnętrzne większości strzelców można przedstawić za pomocą stosunkowo małych wykresów wyszukiwania ścieżek, które można szybko przeszukiwać. Rebellion używał tego samego systemu punktów orientacyjnych do odnajdywania ścieżek i taktycznej sztucznej inteligencji w Sniper Elite, podczas gdy Monolith stworzył zupełnie inną reprezentację dla No One Lives Forever 2. W rozwiązaniu Monolith obszar, do którego postać mogła się przenieść, był reprezentowany przez nakładające się ", który następnie utworzył wykres odnajdywania ścieżki. Punkty orientacyjne systemu działania nie brały bezpośredniego udziału w pathfindingu (poza celami, które pathfinder mógł planować).

Wcześniej twórcy używali szeregu przedstawień do odnajdywania ścieżki. Od tego czasu stało się prawie (ale nie całkiem) wszechobecne używanie siatek nawigacyjnych do reprezentowania przestrzeni wewnętrznych. Większym wysiłkiem jest ujednolicenie podejścia opartego na siatce nawigacyjnej z solidną analizą taktyczną i nierzadko zdarza się, że analiza taktyczna oparta na siatce przebiega równolegle z siatkami nawigacyjnymi do znajdowania ścieżek. Istnieją jednak jeszcze inne realne podejścia. Wolumeny odnajdywania ścieżek w Monolith to kolejne podejście, a wiele gier rozgrywanych na zewnątrz nadal opiera się na wykresach odnajdywania ścieżek opartych na siatce. Gry rozgrywane głównie w pomieszczeniach w naturalny sposób dzielą swoje poziomy na sektory, często oddzielone portalami (technologia optymalizacji renderowania). Sektory te mogą w naturalny sposób działać jako wykres wyszukiwania ścieżek wyższego poziomu do planowania tras długodystansowych. To sprawia, że hierarchiczne algorytmy odnajdywania ścieżek w naturalny sposób pasują do implementacji zdolnych do radzenia sobie z dużymi poziomami.

STRZELANKI

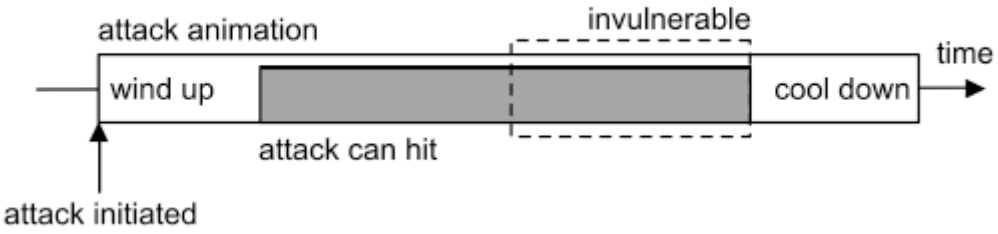
Różne gry wykorzystują punkt widzenia pierwszej lub trzeciej osoby z postaciami podobnymi do ludzi. Gracz bezpośrednio kontroluje jedną postać, używaną jako punkt widzenia gry, a wrogie postacie mają zazwyczaj podobne zdolności fizyczne. W połączeniu z naturalnym konserwatyzmem ustawień gry oznacza to, że wiele gatunków, których nie można nazwać strzelankami, wykorzystuje bardzo podobne techniki sztucznej inteligencji. Dlatego mają zwykle tę samą podstawową architekturę. Zamiast ponownie omawiać ten sam obszar, rozważę te gatunki pod kątem tego, co dodają lub usuwają z podstawowej konfiguracji strzelanki.

Gry platformowe i przygodowe

Gry platformowe są zwykle przeznaczone dla młodszych odbiorców niż strzelanki FPS. Głównym celem projektu jest sprawienie, aby postacie wroga były interesujące, ale dość przewidywalne. Powszechne jest obserwowanie oczywistych wzorców zaprojektowanych w zachowaniu postaci. Gracz jest nagradzany za obserwowanie poczynań wroga i budowanie pomysłu na wykorzystanie jego słabości. To samo dotyczy gier przygodowych, w których wrogowie stają się kolejną zagadką do rozwiązania. Na przykład w Beyond Good and Evil Sekcje Alfa, skądinąd niewrażliwy wróg, opuszczają tarcze na kilka sekund po ataku. W obu przypadkach sztuczna inteligencja użyje podobnych, ale prostszych technik do tych obserwowanych w strzelankach. Ruch zazwyczaj wykorzystuje to samo podejście, chociaż gry platformowe często dodają latających wrogów, których trzeba będzie kontrolować za pomocą algorytmów ruchu 2½D lub 3D. W szczególności gry przygodowe kładą większy nacisk na animację, aby komunikować działania postaci. Niewielka liczba gier pozwala ich postaciom odnaleźć drogę. Nawet w Jak and Daxter: The Precursor Legacy [148] czasami używano nawigacji. Ta gra przyjęła reprezentację siatki nawigacyjnej, aby umożliwić postaciom inteligentne poruszanie się. W wielu grach platformowych ruch może być bezpiecznie utrzymywany lokalnie. Stan wiedzy w zakresie podejmowania decyzji to wciąż najprostsze techniki. Zazwyczaj postacie mają dwa stany: stan „zauważenia gracza" i stan „normalnego zachowania". Normalne zachowania często ograniczają się do stania i odtwarzania wybranych animacji lub ustalonych tras patroli. Na przykład w serii Oddworld [156] niektóre zwierzęta poruszają się losowo, wykorzystując różne zachowania wędrowne, dopóki nie zauważą bohatera. Kiedy postać zauważy gracza, zazwyczaj namierza gracza, zachowując się w poszukiwaniu lub ściganiu. W niektórych grach to naprowadzanie ogranicza się do celowania w gracza i poruszania się do przodu. Inne gry rozszerzają możliwości poruszającej się postaci. Na przykład ludzcy wrogowie w Tomb Raider III] i późniejszych grach z tej serii chwytają się i wspinają na bloki, aby dostać się do Lary. Zwiększyło się to w każdej grze z serii Dark Souls. Różni wrogowie w Dark Souls 3 mogą poruszać się daleko po poziomie, gdy są zaatakowani (tj. W stanie „domu na wrogu"), korzystając z różnych cech otoczenia, w tym drabin i półek jednokierunkowych. I, w przeciwieństwie do poprzednich gier z tej serii, jest mniej prawdopodobne, że utkną lub spadną z krawędzi. Oczywiście istnieją wariacje na ten temat: niektóre postacie mogą mieć kilka innych stanów, mogą wołać o pomoc, mogą występować różne działania w zwarciu i na odległość i tak dalej. Ale nie przychodzi mi do głowy żadna gra w tych gatunkach, w której postacie używają zasadniczo bardziej złożonych technik. To prawdopodobnie nie przypadek: większa złożoność byłaby dla gracza trudna do interpretacji i może wydawać się niesprawiedliwa. Często preferowane jest zachowanie przewidywalne, nawet przy ryzyku wiarygodności.

WALKA W ZWARCIU

W poprzedniej części wspomniałem o Dark Souls i jego kontynuacjach, grze, która nie mieści się w części poświęconej strzelankom. Chociaż istnieje wiele podobieństw między strzelankami a grami akcji w zwarciu, jeśli chodzi o sztuczną inteligencję postaci, walka wręcz różni się zasadniczo od używania broni palnej. Mechanika walki wręcz waha się od prostych do złożonych. Prostym końcem są nieprzerywalne akcje ataków, które kończą się sukcesem, gdy postać znajduje się w zasięgu broni, prawdopodobnie z pewną losową szansą w zależności od statystyki tarczy ofiary lub uników. Na skrajnej złożoności znajdują się gry walki, które mogą zawierać oszałamiający wachlarz mechanik, w tym anulowania, odwrócenia, kombinacje i promocje. Poza najprostszymi grami, walka w zwarciu jest zasadniczo i wyjątkowo związana z wyczuciem czasu. Rysunek pokazuje schematyczne przedstawienie przykładowego ruchu w systemie walki wręcz.



Ma kilka faz:

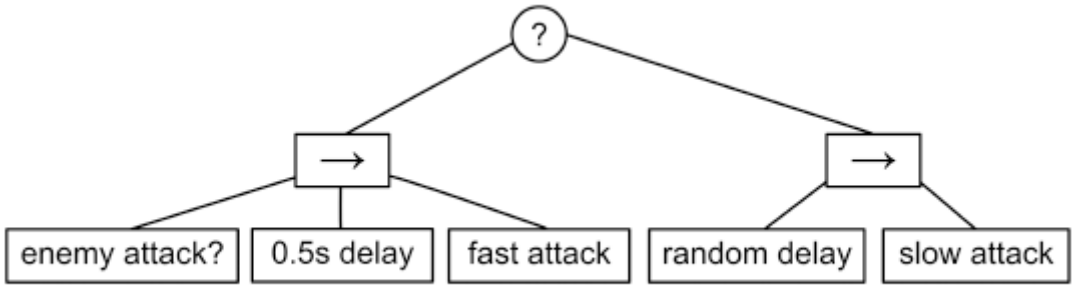
1. Okres nakręcania, kiedy animacja ataku rozpoczęła się, ale nie może zadać obrażeń, a postać nadal jest podatna na ataki.

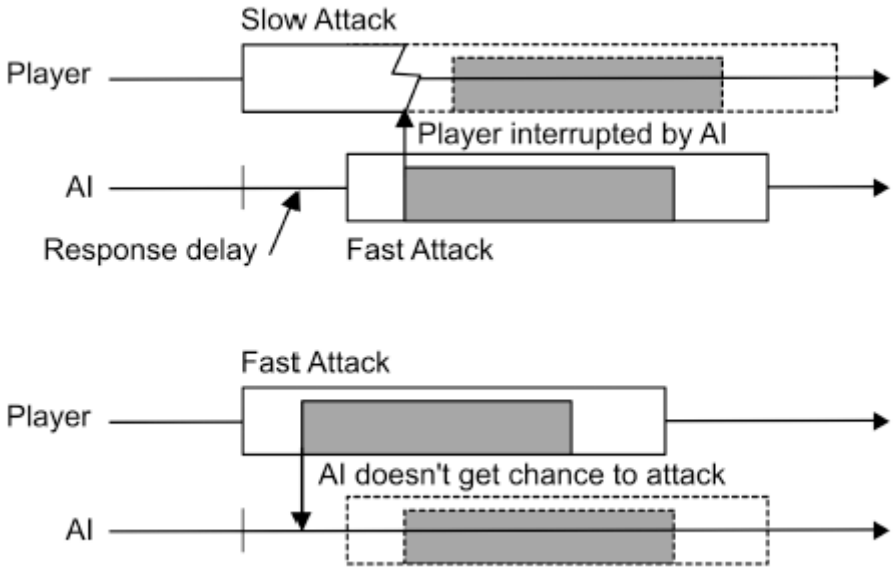
2. Okres, w którym postać jest w stanie zadać obrażenia, ale atak może zostać przerwany lub sparowany przez wroga.

3. Niezwyciężony okres, w którym atak może zadać obrażenia, ale nie można go przerwać.

4. Czas odnowienia, kiedy atak nie jest już niebezpieczny, ale gracz nie może rozpocząć kolejnej akcji.

Działa tu kilka kryteriów: czy animacja jest uruchomiona, a akcja jest w toku; czy gracz jest podatny na atak, przerwanie lub sparowanie; czy atak jest w stanie zranić przeciwnika; oraz czy gracz może zakończyć aktualny atak, aby rozpocząć kolejną akcję. Rysunek pokazuje jeden przykład tego, jak można je ułożyć, ale każda gra będzie je sekwencjonować na różne sposoby, często z różnymi postaciami, a nawet różnymi broniami mającymi własny wzór. W prawie wszystkich przypadkach, nawet gdy fazy są takie same, czas będzie się różnić. Sztuczna inteligencja w tych grach musi podejmować decyzje nie tylko o tym, który ruch wykonać, ale także kiedy. Nie wymaga to unikalnego algorytmu, można użyć tych samych narzędzi decyzyjnych, które widzieliśmy w tej książce, ale ich zawartości (na przykład konkretne wzorce stanu w maszynie stanów lub węzły w drzewie zachowań). muszą być zaprojektowane w taki sposób, aby uwzględnić czas. Rysunek pokazuje przykład części drzewa zachowań, a drugi rysunek pokazuje drzewo w użyciu, ze schematem czasowym pomyślnie obronionego ataku i ataku, który przechodzi.





W przygodowych grach akcji z walką wręcz częścią rozwoju umiejętności gracza jest zrozumienie zestawu ruchów wroga i tego, jak każdy z nich zareaguje na działania gracza. W tym przypadku korzystny jest prosty algorytm podejmowania decyzji. Gracz dowie się, że dany wróg dobrze paruje powolne ciosy mieczem, o ile jego reakcje są względnie spójne. Oczywiście może być zbyt zautomatyzowany. Pewna randomizacja jest prawdopodobnie konieczna, więc na przykład paruje tylko cztery razy na pięć. Można to jednak osiągnąć za pomocą drzew decyzyjnych, maszyn stanowych lub drzewek zachowań — prostych podejść do podejmowania decyzji, które pozwolą graczowi „nauczyć się" ataków wroga. Nawet przy użyciu prostych narzędzi można być wysoce wyrafinowanym i przebiegłym dzięki projektowi AI. W rzeczywistości wdrożenie sztucznej inteligencji jest stosunkowo proste, która jest nie do pokonania, zdolna do reagowania doskonale do każdego ataku w ramach jego zainicjowania. Wiele trudności we wdrażaniu tego rodzaju sztucznej inteligencji polega na odpowiednim wyczuciu. Spacer po cienkiej granicy między wyzwaniem a uczciwością. W tym miejscu sztuczna inteligencja staje się bardziej sztuką niż nauką, a żadna technika nie uchroni cię przed ulepszaniem i testowaniem rozgrywki.

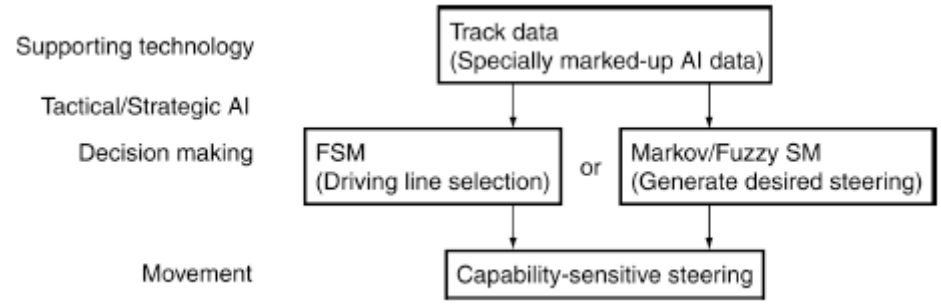
MMOG

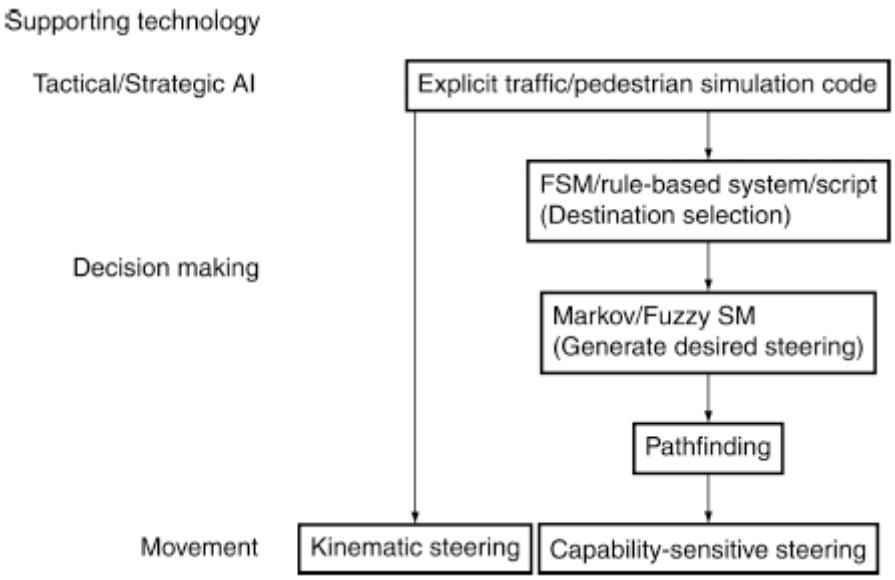
Gry online dla wielu graczy (MMOG) zwykle angażują dużą liczbę graczy w trwałym świecie. Technicznie ich najważniejszą cechą jest oddzielenie serwera, na którym uruchomiona jest gra, od maszyn, na których gracz gra. Rozróżnienie między klientem a serwerem jest zwykle wprowadzane w strzelankach (i wielu innych typach gier), aby ułatwić programowanie trybów dla wielu graczy. Jednak w grze MMOG serwer nigdy nie będzie działał na tej samej maszynie co klient; to będzie normalnie działać na zestawie dedykowanego sprzętu. Możemy zatem wykorzystać więcej zasobów pamięci i procesora. W niektórych grach dla wielu graczy istnieje tylko marginalne zapotrzebowanie na sztuczną inteligencję. Jedynymi postaciami kontrolowanymi przez AI są zwierzęta lub dziwny potwór. Wszystkie postacie w grze grane są przez ludzi. Chociaż może to być idealna sytuacja, nie zawsze jest to praktyczne. Gra wymaga pewnej masy krytycznej graczy, zanim będzie warta poświęcenia na nią czasu. Większość gier MMOG dodaje do gry wyzwanie oparte na sztucznej inteligencji, podobnie jak w każdej przygodzie z perspektywy pierwszej lub trzeciej osoby. Przy tak ogromnym świecie gry wszystkie wyzwania stojące przed twórcą sztucznej inteligencji pojawiają się pod względem skali. Zastosowane technologie są w dużej mierze takie same jak w strzelance, ale ich implementacja musi być znacząco różna, aby poradzić

sobie z dużą liczbą postaci i znacznie większym światem. Podczas gdy prosty pathfinder A * poradzi sobie z poziomem w strzelance i 5 do 50 postaciami, które używają go do planowania tras, prawdopodobnie zatrzyma się, gdy 1000 postaci będzie musiało zaplanować swoją drogę po świecie wielkości kontynentu. To właśnie te wielkoskalowe technologie, w szczególności odnajdywanie ścieżek i percepcja sensoryczna, wymagają bardziej skalowalnych wdrożeń. Przyjrzeliśmy się niektórym z nich. Na przykład w odnajdywaniu ścieżek możemy łączyć planistów, używać hierarchicznego odnajdywania ścieżek lub używać geometrii instancji.

DRIVING

Prowadzenie pojazdu to jedno z najbardziej wyspecjalizowanych zadań związanych ze sztuczną inteligencją, charakterystycznych dla danego gatunku, dla programisty. W przeciwieństwie do innych gatunków, kluczowe zadania AI koncentrują się wokół ruchu. Zadaniem nie jest stworzenie realistycznego zachowania w poszukiwaniu celu, sprytnego rozumowania taktycznego ani znajdowania trasy, chociaż wszystko to może wystąpić w niektórych grach samochodowych. Gracz oceni kompetencje sztucznej inteligencji na podstawie tego, jak dobrze prowadzi samochód. Rysunek pokazuje architekturę AI dopasowaną do gry wyścigowej, a drugi rysunek rozszerza tę architekturę do użytku w grze miejskiej, w której możliwe są różne trasy, a pojazdy w otoczeniu współdzielą drogę.





RUCH

W przypadku gier wyścigowych programista ma dwie opcje, które zaimplementują ruch samochodu. Najprostszym podejściem jest umożliwienie projektantowi poziomów stworzenie jednej lub więcej linii

wyścigowych, wzdłuż których pojazd może osiągnąć optymalną prędkość. Ta linia wyścigowa może być następnie sztywno śledzona. Może to w ogóle nie wymagać sterowania. Samochody sterowane komputerowo mogą po prostu poruszać się po określonej ścieżce. Zazwyczaj ten rodzaj linii wyścigowej jest definiowany jako spline: krzywa matematyczna. Splajny są definiowane jako krzywe w przestrzeni, ale mogą również zawierać dodatkowe dane. Dane dotyczące prędkości zawarte w splajnie umożliwiają sztucznej inteligencji dokładne sprawdzenie położenia i prędkości samochodu w dowolnym momencie i odpowiednie ich renderowanie. Zapewnia to bardzo ograniczony system: samochody nie mogą łatwo wyprzedzać się nawzajem, nie unikną kolizji przed nimi i nie zostaną odchylone podczas zderzenia z graczem. Aby uniknąć tych oczywistych ograniczeń, dodawany jest dodatkowy kod, aby upewnić się, że jeśli samochód zostanie wybity z pozycji, można zastosować proste zachowanie kierownicy, aby przywrócić go na linię wyścigu. Wciąż charakteryzuje się tendencją samochodów do wpadania do rozbitego samochodu z naiwnym zapałem. Większość wczesnych gier samochodowych, takich jak Formuła 1 [84], wykorzystywała to podejście. Był również używany w wielu ostatnich grach do kontrolowania samochodów, które mają być częścią „tła", jak widać w Grand Theft Auto 3 [104]. Drugie podejście, stosowane w przeważającej mierze w ostatnich tytułach, polega na tym, aby sztuczna inteligencja kierowała samochodem — aby zastosować dane sterujące do symulacji fizyki, aby samochód zachowywał się realistycznie. Stopień, w jakim fizyka, z którą samochody AI muszą sobie radzić, jest taka sama jak fizyka, której doświadcza gracz, jest kwestią krytyczną. Zazwyczaj gracz ma nieco trudniejszą fizykę niż samochody sterowane przez sztuczną inteligencję, chociaż wiele gier daje teraz sztucznej inteligencji to samo zadanie, co gracz. Bardzo często zdarza się, że w tego typu grach są definiowane linie wyścigowe. Samochód sterowany przez sztuczną inteligencję stara się podążać za linią wyścigową, prowadząc samochód, zamiast sprawiać, że linia wyścigowa działa jak szyna, po której może się poruszać. Oznacza to, że sztuczna inteligencja często nie może osiągnąć pożądanej linii, zwłaszcza jeśli została popchnięta przez inny samochód. Może to spowodować dodatkowe problemy. W serii Gran Turismo „symulatora jazdy", która wykorzystuje to podejście, samochód może zostać zepchnięty z pozycji przez gracza. W tym momencie samochód nadal próbowałby jechać po torze wyścigowym, co zwykle skutkowało rozbiciem się na następnym zakręcie i wpadnięciem w żwirową pułapkę. Aby rozwiązać problem wyprzedzania, gdy wolniej poruszający się pojazd znajduje się na torze wyścigowym, wielu programistów dodaje specjalne zachowania związane z kierowaniem: samochód będzie czekał na długą prostą, a następnie wycofał się, aby wyprzedzić. Jest to charakterystyczne zachowanie podczas wyprzedzania obserwowane w wielu grach wyścigowych od Gran Turismo po Burnout i jest częstą sztuczką wyprzedzania w rzeczywistych wyścigach samochodami o średniej i niskiej mocy. Jednak większość wyprzedzania w najszybszych seriach wyścigowych na świecie (takich jak Formuła 1) odbywa się podczas hamowania na zakrętach. Można to osiągnąć za pomocą alternatywnej linii wyścigowej zdefiniowanej przez projektanta poziomów. Jeśli samochód chce wyprzedzać, zajmuje pozycję na tej linii, co zapewni mu możliwość późniejszego hamowania i przejęcia kontroli nad wyjściem z zakrętu. W prawdziwych wyścigach kierowcy mogą podjąć pewne działania obronne, aby zablokować potencjalne wyprzedzenia (choć może to być ograniczone przez zasady serii). To sprawia, że takie alternatywne linie wyścigowe są trudne do zdefiniowania, szczególnie gdy gracz jest wyprzedzany. Na szczęście, mniej skuteczna sztuczna inteligencja w tym momencie sprawia, że gracz czuje się bardziej uzdolniony, a zatem jest mniej prawdopodobne, że zostanie negatywnie oceniony.

Odmiana podejścia polegająca na podążaniu za linią wyścigową jest używana w wielu grach rajdowych i jest czasami nazywana „goń za królikiem". Niewidzialny cel (tytułowy królik) porusza się wzdłuż linii wyścigu przy użyciu metody bezpośredniej aktualizacji pozycji. Pojazd sterowany przez sztuczną inteligencję po prostu celuje w królika; można nim sterować na przykład za pomocą zachowania „przybycia". Ponieważ królik jest zawsze trzymany przed samochodem, zaczyna skręcać pierwszy,

upewniając się, że samochód skręca we właściwym miejscu. Jest to szczególnie przydatne w grach rajdowych, ponieważ sprawia, że wykonywanie poślizgów mocy jest całkiem naturalne. Samochód automatycznie zacznie skręcać na długo przed zakrętem, a jeśli zakręt będzie ostry, będzie skręcał mocno, powodując, że symulacja fizyki pozwoli trochę wyślizgnąć się tyłowi samochodu. Inni programiści używali narzędzi do podejmowania decyzji w ramach napędzającej AI. Symulator kartingowy Manic Karts wykorzystywał rozmyte podejmowanie decyzji zamiast linii wyścigowych. Określił lewy i prawy zakres toru w niewielkiej odległości przed pojazdem, a także pobliskie gokarty, a następnie użył odręcznie napisanego automatu stanu Markowa, aby określić, co dalej robić. Forza Motorsport [188] wykorzystał sieci neuronowe, aby nauczyć się prowadzić, obserwując ludzi. Ostateczna sztuczna inteligencja dostarczona z grą była wynikiem setek godzin szkolenia zespołu programistów. Do niedawna takie techniki były rzadkością. Nie były one na tyle szeroko stosowane, by sugerować, że zapewniały radykalnie lepszą wydajność. Jednak dzięki niedawnym dramatycznym sukcesom w stosowaniu głębokiego uczenia w rozgrywce (na przykład [43]), sieci neuronowe cieszą się powszechnym renesansem w branży i są wykorzystywane przy tworzeniu kilku nadchodzących gier wyścigowych. W ciągu najbliższych pięciu lat zobaczymy, czy z entuzjazmu przerodzi się w wszechobecność.

ODKRYWANIE ŚCIEŻEK I TAKTYCZNA AI

Wraz z Driver pojawił się nowy gatunek gier samochodowych. Tutaj nie ma ustalonego toru. Akcja gry toczy się na ulicach miast, a jej celem jest łapanie lub unikanie innych samochodów. Samochód może jechać dowolną trasą, a uciekając przed policją, gracz zwykle skręca i cofa. Pojedynczy, stały tor wyścigowy nie ma zastosowania do tego rodzaju gry. W wielu grach z tego gatunku sztuczna inteligencja wroga podąża wyznaczoną ścieżką, gdy ucieka przed graczem lub wykonuje prosty algorytm naprowadzania, gdy próbuje go złapać. W Grand Theft Auto 3 i jego kontynuacjach samochody są tworzone tylko dla kilku bloków otaczających pozycję gracza. Kiedy policja dotrze do gracza, są zbierani z tego obszaru, a dodatkowe samochody są wstrzykiwane w odpowiednie miejsca. Ponieważ tego rodzaju gra symuluje większy obszar, pojazdy zaczynają potrzebować odnajdywania ścieżki, aby znaleźć swoją trasę, szczególnie w celu schwytania gracza. To samo dotyczy wykorzystania analizy taktycznej do opracowania prawdopodobnych dróg ucieczki i ich blokowania. Kierowca używa prostego algorytmu, aby otoczyć gracza. Analiza taktyczna oparta na aktualnym kierunku, w którym porusza się gracz, może następnie poprosić AI wozu policyjnego o przechwycenie. Radiowozy mogą następnie korzystać z taktycznego odnajdywania drogi, aby dostać się na swoje pozycje bez przekraczania ścieżki gracza (aby uniknąć oddania gry).

GRY JAK JAZDA

Podstawowe podejście stosowane do prowadzenia gier może mieć zastosowanie do wielu innych gatunków. Niektóre gry o sportach ekstremalnych, takie jak zręcznościowa seria SSX oraz nowsza i przypominająca symulację Steep , mają mechanikę wyścigową (w tym ostatnim przypadku między innymi trybami). Nałożony na system wyścigowy (zwykle implementowany przy użyciu tej samej sztucznej inteligencji opartej na liniach wyścigowych, co w przypadku gier samochodowych) jest zwykle pod-grą „sztuczek", która polega na planowaniu animowanych sztuczek podczas skoków. Można je dodać w predefiniowanych punktach na linii wyścigu (tj. znacznik, który mówi, że gdy postać osiągnie ten punkt, zaplanuj trik o określonym czasie trwania) lub mogą być wykonane przez system podejmowania decyzji, który przewiduje prawdopodobny czas antenowy, który będzie wynik i planuje trik o odpowiednim czasie trwania. Futurystyczne wyścigówki, takie jak Wipeout i jego sequele, również opierają się na tej samej technologii wyścigowej AI. Powszechne jest, że tego rodzaju gry zawierają broń. Aby to wesprzeć, potrzebna jest dodatkowa architektura AI obejmująca celowanie

(często nie jest to pełne rozwiązanie strzelania, ponieważ broń trafia w cel) i podejmowanie decyzji (pojazd może zwolnić, aby umożliwić wrogowi wyprzedzenie go w celu namierzenia ich).

STRATEGIA CZASU RZECZYWISTEGO

Dzięki Dune II2 Westwood stworzył nowy gatunek, który stał się podstawą portfolio wydawców. Mimo że stanowi niewielką część całkowitej sprzedaży gier, gatunek ten jest jednym z najsilniejszych na platformie PC. Kluczowe wymagania AI dla gier strategicznych czasu rzeczywistego to:

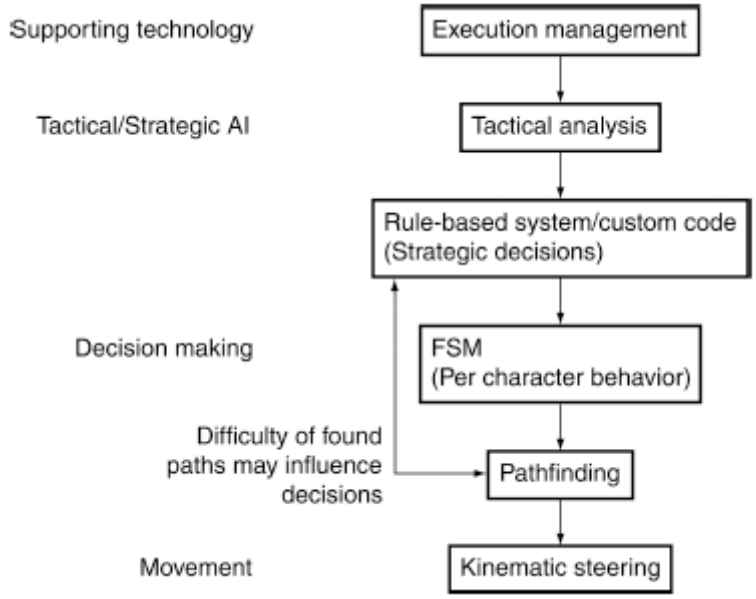
• Znalezienie drogi

• Ruch grupowy

• Taktyczna i strategiczna sztuczna inteligencja

• Podejmowanie decyzji

Rysunek przedstawia architekturę AI dla gry strategicznej czasu rzeczywistego (RTS).



Różni się to bardziej w zależności od gry niż w poprzednich gatunkach, w zależności od konkretnego zestawu elementów rozgrywki. Poniższy model powinien służyć jako przydatny punkt wyjścia dla twojego własnego rozwoju.

ODKRYWANIE ŚCIEŻEK

Wczesne gry strategiczne czasu rzeczywistego, takie jak Warcraft: Orcs and Humans i Command and Conquer , były synonimem algorytmów odnajdywania ścieżek, ponieważ skuteczne odnajdywanie ścieżek było głównym wyzwaniem technicznym sztucznej inteligencji. Przy dużych poziomach opartych na siatce (często obejmujących dziesiątki tysięcy pojedynczych kafelków), długich problemach ze znajdowaniem ścieżek (gracz może wysłać jednostkę po całej mapie) i wielu dziesiątkom jednostek, szybkość odnajdywania ścieżek ma kluczowe znaczenie. Chociaż większość gier nie używa już grafiki opartej na kafelkach, podstawowa reprezentacja jest nadal oparta na siatce. Większość gier używa zwykłej tablicy wysokości (zwanej polem wysokości) do renderowania krajobrazu. Ta sama tablica jest następnie używana do wyszukiwania ścieżek, dając regularną strukturę opartą na siatce. Niektórzy programiści wstępnie obliczają dane tras dla wspólnych ścieżek na każdym poziomie. Na niektórych poziomach StarCraft II zniszczalny teren może zmienić wykres nawigacji podczas rozgrywki, chociaż

zazwyczaj zmiana jest stosunkowo niewielka: dodano lub usunięto garść połączeń. Gry takie jak Company of Heroes wprowadziły graczy w całkowicie odkształcalny teren, gdzie wyczerpujące wstępne obliczenia są trudne.

RUCH GRUPOWY

Gry takie jak Kohan: Ahriman's Gift i Warhammer: Dark Omen grupowały poszczególne osoby w zespoły i zmuszały je do poruszania się jako całość. Osiąga się to za pomocą systemu ruchu formacji z predefiniowanymi wzorcami. W Homeworld i jego kontynuacjach formacje są rozszerzane na trzy wymiary, dając wrażenie lotu w kosmos mimo utrzymywania silnego kierunku w górę iw dół. Tam, gdzie formacje Kohana mają ograniczony rozmiar, w Homeworld może brać udział dowolna liczba jednostek. Wymaga to skalowalnych formacji z różnymi pozycjami szczelin dla różnej liczby jednostek. Większość gier RTS używa teraz jakiegoś rodzaju formacji. Prawie wszystkie z nich definiują formacje w kategoriach ustalonego wzoru (przy ustalonym zestawie znaków w formacji), który porusza się jako całość. W Full Spectrum Warrior (kolejnej grze typu RTS, która opisuje się inaczej), formacja zależy od cech otaczającego ją poziomu. Przy murze oddział ustawia się w jednej linii, za przeszkodą dającą osłonę, zwija się, a na otwartej przestrzeni tworzą klin. Gracz ma jedynie pośrednią kontrolę nad kształtem formacji. Gracz kontroluje, dokąd zmierza drużyna, a sztuczna inteligencja określa wzór formacji, którego należy użyć. Gra jest również niezwykła, ponieważ jej formacje kontrolują ostateczną lokalizację postaci dopiero po ich przesunięciu. Podczas ruchu jednostki poruszają się niezależnie i mogą zapewnić sobie wzajemną osłonę na żądanie.

TAKTYCZNA I STRATEGICZNA AI

Jeśli wczesne gry RTS były pionierami sztucznej inteligencji w grach, wykorzystując odnajdywanie ścieżek, to gry pod koniec lat 90. robiły to samo z taktyczną sztuczną inteligencją. Mapowanie wpływów zostało opracowane do użytku w grach RTS i dopiero niedawno zaczęło być interesujące dla innych gatunków (zwykle w formie taktyki punktów orientacyjnych). Do tej pory wyniki taktycznej i strategicznej sztucznej inteligencji były głównie wykorzystywane do kierowania odnajdywaniem ścieżek. Wczesnym przykładem była Total Annihilation, gdzie jednostki uwzględniają złożoność terenu podczas opracowywania ścieżek; prawidłowo poruszają się po wzgórzach lub innych formacjach skalnych. Ta sama analiza służy również do kierowania strategicznych decyzji w grze. Drugim powszechnym zastosowaniem jest wybór lokalizacji pod budowę. Dzięki mapie wpływów pokazującej obszary pod kontrolą znacznie łatwiej jest bezpiecznie zlokalizować ważny obiekt budowlany. Podczas gdy jeden budynek zajmuje tylko jedną lokalizację, ściany są częstą cechą wielu gier RTS i są trudniejsze w obsłudze. Na przykład ściany w Warcraft zostały zbudowane wcześniej przez projektanta poziomów. W Empire Earth [179] sztuczna inteligencja była odpowiedzialna za budowę murów, wykorzystując kombinację mapowania wpływów i rozumowania przestrzennego (AI próbowała umieszczać mury między budynkami ekonomicznie wrażliwymi a prawdopodobnymi pozycjami wroga). Dużo mówiło się w kręgach AI w grach o używaniu analizy taktycznej do planowania manewrów wojsk na dużą skalę – na przykład wykrywaniu słabych punktów w formacji wroga – i rozmieszczaniu jednostek całej strony, aby to wykorzystać. Do pewnego stopnia odbywa się to w każdej grze RTS: AI będzie kierować jednostki tam, gdzie myśli, że jest wróg, zamiast po prostu zmieść je na mapie w losowe miejsce. Jest to kontynuowane w grach takich jak Empire: Total War [186], w których sztuczna inteligencja spróbuje manewrować poza zasięgiem broni rakietowej i armat przed rozpoczęciem ataków na wiele flanków. Jest to jeszcze trudniejsze w poziomach reprezentujących bitwy morskie, w których ważnym czynnikiem jest przeważający wiatr. Istnieje potencjał, aby pójść jeszcze dalej i mieć sztuczną inteligencję uzasadnienie możliwych strategii ataku w świetle analizy taktycznej i tras, które każda jednostka musiałaby obrać, aby wykorzystać wszelkie słabości. Widziałem kilka przykładów gier, które ewidentnie zaszły tak daleko. Ponieważ analiza taktyczna jest tak mocno powiązana z grami RTS,

dyskusja w Części 6 była ukierunkowana na ten gatunek. Pozostaje tylko przeanalizować zachowanie, którego oczekujesz od strony kontrolowanej przez komputer i wybrać odpowiedni zestaw analiz do wykonania.

PODEJMOWANIE DECYZJI

Istnieje kilka poziomów, na których podejmowanie decyzji musi odbywać się w grze RTS, więc prawie zawsze wymagają one wielopoziomowego podejścia AI. Niektóre proste decyzje są często wykonywane przez poszczególne postacie. Na przykład łucznicy w Warcraft podejmują własne decyzje, czy utrzymać swoją lokalizację, czy ruszyć do przodu, aby zaatakować wroga. Na poziomie średniozaawansowanym formacja lub grupa postaci może wymagać podjęcia pewnych decyzji. W Full Spectrum Warrior cała drużyna może podjąć decyzję o ukryciu się, gdy zostanie wystawiona na ostrzał wroga. Ta decyzja przechodzi następnie na każdą indywidualną postać, aby zdecydować, jak najlepiej się ukryć (na przykład leżeć na ziemi). Większość trudnych decyzji podejmowanych jest na poziomie całej drużyny w grze. Zazwyczaj w tym samym czasie będzie się działo wiele różnych rzeczy: właściwe zasoby wymagają zebrania, badania muszą być pokierowane, budowa powinna być zaplanowana, jednostki muszą być wyszkolone, a siły muszą być zebrane do obrony lub ataku. Dla każdego z tych wymagań tworzony jest komponent AI. Złożoność tego jest bardzo zróżnicowana w zależności od gry. Aby ustalić kolejność badań, moglibyśmy na przykład użyć punktacji liczbowej dla każdego postępu i wybrać następny postęp o najwyższej wartości. Alternatywnie moglibyśmy mieć algorytm wyszukiwania, taki jak Dijkstra, aby wypracować najlepszą ścieżkę od obecnego zestawu znanych technologii do technologii celu. W grach takich jak Warcraft każdy z tych modułów AI jest w dużej mierze niezależny. Sztuczna inteligencja, która planuje zbieranie zasobów, nie planuje z wyprzedzeniem zgromadzić określonego zasobu na późniejsze prace budowlane. Po prostu przypisuje zrównoważony wysiłek w celu zebrania dostępnych zasobów. Wojskowe dowództwo AI również czeka, aż zgromadzi się wystarczająca ilość sił, zanim zaatakuje wroga. Gry takie jak Warcraft 3: Reign of Chaos wykorzystują centralnie sterującą sztuczną inteligencję, która może wpływać na niektóre lub wszystkie moduły. W tym przypadku ogólna sztuczna inteligencja może zdecydować, że chce grać w ofensywną grę, i w tym celu zniekształci wysiłek konstrukcyjny, szkolenie jednostek i sztuczną inteligencję wojskową. W grach RTS różne poziomy AI są często nazywane stopniami wojskowymi. Dowodzić będzie generał lub pułkownik, a niżej możemy mieć dowódców lub poruczników, aż do pojedynczych żołnierzy. Chociaż to nazewnictwo jest powszechne, prawie nie ma zgody co do tego, jak powinien nazywać się każdy poziom, co może być bardzo mylące. W jednej grze całe przedstawienie może kontrolować ogólna sztuczna inteligencja. W innej grze jest to po prostu sztuczna inteligencja odpowiedzialna za działania militarne, pod przewodnictwem króla lub prezydenta AI. Wybór technologii podejmowania decyzji odzwierciedla to, co w innych grach. Zazwyczaj większość decyzji podejmowanych jest za pomocą prostych technik, takich jak maszyny stanów i drzewa decyzyjne Markowa lub inne metody probabilistyczne są bardziej powszechne w grach RTS niż w innych gatunkach. Podejmowanie decyzji na potrzeby rozmieszczenia wojskowego jest często prostym zestawem reguł (czasami systemem opartym na regułach, ale zwykle zakodowanymi na stałe instrukcjami IF-THEN) opartymi na wynikach silnika analizy taktycznej.

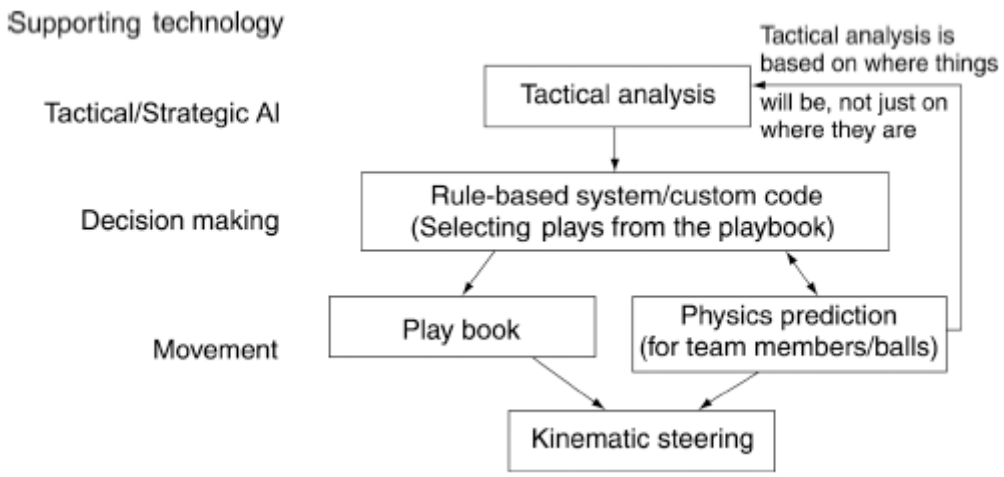
MOBA

Internetowe areny bitewne dla wielu graczy zyskały na znaczeniu na początku 2010 roku, stając się jednym z najważniejszych gatunków gier. To, co początkowo zaczęło się jako „Defense of the Ancients" - mod/mapa generowana przez graczy do Warcraft III - zyskało na znaczeniu wraz z głównym konkurentem League of Legends i opracowanym przez studio sequelem, Dota 2 [195]. Pomimo pośpiechu w rywalizacji w tym popularnym nowym gatunku, niewiele innych gier odniosło znaczący sukces komercyjny. Chociaż gorączka złota przeszła (w szczególności do strzelanek opartych na bohaterach, a następnie do szału Battle Royale), w momencie pisania tego tekstu gry MOBA są nadal

najważniejszym gatunkiem gier pod względem e-sportu. Spośród 20 turniejów e-sportowych z najwyższymi nagrodami pieniężnymi, 18 to turnieje typu MOBA. Geneza tego gatunku w strategii czasu rzeczywistego przenosi się na sztuczną inteligencję. W grze występują dwa rodzaje postaci: bohaterowie i creepy (w zależności od gry mogą być również nazywane sługami lub mobami). Bohaterowie mają być kontrolowani przez gracza. Boty AI zostały opracowane, aby nimi grać, ale nie są jeszcze konkurencyjne z ludzkimi graczami. Creepy są porównywalne z pojedynczymi jednostkami w grze RTS. Poruszają się po ustalonej linii lub czają się gdzieś w pozostałej części mapy (znanej jako dżungla w Dota i League of Legends). Creepy z linii zazwyczaj należą do jednej z drużyn graczy i będą atakować tylko członków drugiej drużyny (bohaterów lub creepów). Pełzacze z dżungli zaatakują każdego, zarówno w zasięgu wzroku, jak i gdy zostaną zaatakowane. Gdy są w stanie ataku, skrada się w dżungli atakuje najbliższego gracza. Creepy są zaprojektowane tak, aby były proste i łatwe do przewidzenia. Częścią umiejętności w grze jest manipulowanie creepami, przewidywanie ich ataków, przekierowywanie ich agresji. Z tego powodu zwykle stosuje się najprostsze możliwe techniki sztucznej inteligencji. Pełzanie nie powinno zachowywać się w wyrafinowany sposób. Powinny zachowywać się tak, jakby zostały zaimplementowane przez maszynę stanową. Ponieważ ruch creepów jest tak ograniczony (poruszanie się wzdłuż linii lub naprowadzanie na wroga), odnajdywanie ścieżki jest rzadko potrzebne. W grach inspirowanych Dotą drużyny mogą mieć kuriera: autonomiczną jednostkę, która wezwana do bohatera dostarcza dostawców. Ta postać może potrzebować nawigowania przez poziom, ale najczęściej jest zaimplementowana jako jednostka latająca, z minimalnym odnajdywaniem ścieżki. Ogólnie rzecz biorąc, wymagania AI w MOBA są zazwyczaj prostsze niż gry RTS, z których zostały wyprowadzone. Podobnie jak w przypadku gier RTS, w tym samym poziomie może działać wiele creepów. Ale ich sztuczna inteligencja rzadko potrzebuje czegoś bardziej wyrafinowanego niż podstawowe podejmowanie decyzji i sterowanie.

SPORTY

Gry sportowe mogą obejmować od głównych ligowych serii sportowych, takich jak Madden NFL 18 [108], po symulatory bilarda, takie jak Mistrzostwa Świata w Snookera (ostatnia roczna premiera to [102]). Mają tę zaletę, że mają ogromną, łatwo dostępną wiedzę na temat dobrych strategii: profesjonaliści, którzy grają w tę grę. Ta wiedza nie zawsze jest jednak łatwa do zakodowania w grze, a oni stają przed dodatkowym wyzwaniem, jakim jest posiadanie graczy, którzy tego oczekują aby zobaczyć kompetencje na poziomie człowieka. W przypadku sportów zespołowych kluczowym wyzwaniem jest to, aby różne postacie reagowały na sytuację w sposób uwzględniający resztę zespołu. Niektóre sporty, takie jak baseball i piłka nożna, mają bardzo silne schematy zespołowe. Przykładem jest przykład podwójnej gry w baseball w Części 3. Rzeczywista pozycja obrońców będzie zależeć od miejsca uderzenia piłki, ale ogólny wzorzec ruchu jest zawsze taki sam. Dlatego gry sportowe zazwyczaj wykorzystują pewnego rodzaju wielopoziomową sztuczną inteligencję. Istnieje sztuczna inteligencja wysokiego poziomu podejmująca strategiczne decyzje (często przy użyciu pewnego rodzaju uczenia się parametrów lub działania, aby upewnić się, że rzuca wyzwanie graczowi). Na niższym poziomie może istnieć skoordynowany system ruchu, który odtwarza wzorce w odpowiedzi na zdarzenia w grze. Na najniższym poziomie każdy gracz będzie miał własną sztuczną inteligencję, aby określić, jak zmieniać zachowanie w ramach ogólnej strategii. Sporty inne niż drużynowe, takie jak tenis pojedynczy, pomijają warstwę środkową; nie ma zespołu do koordynowania. Rysunek przedstawia architekturę typowej sztucznej inteligencji gry sportowej.



PRZEWIDYWANIE FIZYKI

W wielu grach sportowych piłki poruszają się z dużą prędkością pod wpływem fizyki. Może to być piłka tenisowa, piłka nożna lub bilard. W każdym przypadku, aby sztuczna inteligencja mogła podejmować decyzje (przechwycić piłkę lub wypracować skutki uboczne uderzenia), musimy być w stanie przewidzieć, jak będzie się zachowywać. W grach, w których dynamika piłki jest złożona i stanowi integralną część gry (np. gry bilardowe i gatunki gier w golfa), może być konieczne uruchomienie fizyki w celu przewidzenia wyniku. Dla prostszych dynamik, takich jak baseball czy piłka nożna, można przewidzieć trajektorie piłki. W każdym przypadku proces jest taki sam, jak w przypadku przewidywania pocisków w rozdziale 3. Te same rozwiązania w zakresie strzelania, które stosuje się w broni palnej, można stosować w grach sportowych.

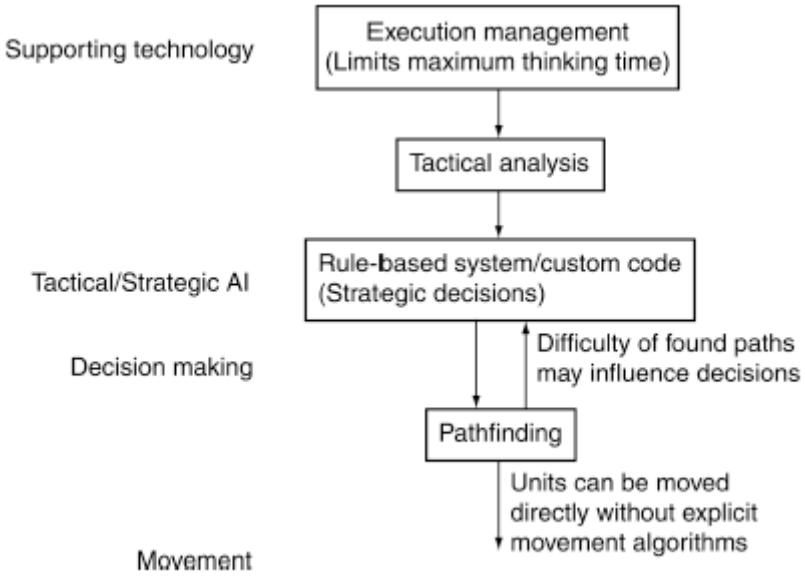
PODRĘCZNIKI I TWORZENIE TREŚCI

Wdrażanie solidnych podręczników jest częstym źródłem problemów związanych ze sztuczną inteligencją w sportach zespołowych. Poradnik składa się z zestawu wzorców ruchowych, z których zespół będzie korzystał w pewnych okolicznościach. Czasami playbook odnosi się do całej drużyny (np. ofensywna gra na linii bójki w piłce nożnej), ale często odnosi się do mniejszej grupy graczy (np. pick-and-roll w koszykówce). Jeśli twoja gra nie zawiera wypróbowanych i przetestowanych zabaw, takich jak ta, dla fanów gry w świecie rzeczywistym będzie oczywiste, że kupią twój produkt. Sekcja skoordynowanego ruchu w rozdziale 3 zawierała algorytmy zapewniające, że postacie poruszają się we właściwym czasie. Zwykle musi to być połączone z systemem ruchu formacji z tego samego część u, aby upewnić się, że członkowie zespołu poruszają się w wizualnie realistycznych wzorach. Oprócz technologii umożliwiającej tworzenie podręczników, należy zadbać o to, aby sztuki mogły być w jakiś sposób napisane. Musi istnieć dobra ścieżka tworzenia treści, aby sztuki mogły wejść do gry. Zazwyczaj jako programista nie znasz wszystkich zagrań, które muszą dotrzeć do finałowej gry, i nie chcesz, aby testowanie każdej kombinacji było obciążone ciężarem. Odsłonięcie formacji i zsynchronizowany ruch są kluczem do umożliwienia ekspertom sportowym tworzenia wzorców do ostatecznej gry.

STRATEGICZNE GRY TUROWE

Turowe gry strategiczne często opierają się na tych samych technikach sztucznej inteligencji, które są używane w grach RTS. Wczesne gry turowe były albo wariantami istniejących gier planszowych (na przykład 3D Tic-Tac-Toe) albo uproszczonymi grami wojennymi (Computer Bismark był jednym z moich wczesnych ulubionych). Oba polegały na technikach minimaksowych używanych do grania w gry planszowe . W miarę jak gry strategiczne stawały się coraz bardziej wyrafinowane, liczba możliwych

ruchów w każdej turze znacznie rosła. W ostatnich grach, takich jak Sid Meier's Civilization VI, gracz ma do dyspozycji niemal nieograniczoną liczbę możliwych ruchów w każdej turze, mimo że każdy ruch jest stosunkowo dyskretny (tj. postać przemieszcza się z jednego miejsca na siatce do drugiego). W grach takich jak seria Worms sytuacja jest jeszcze szersza. Podczas tury gracza przejmują kontrolę nad każdą postacią i poruszają ją w trzecioosobowy sposób (na ograniczony dystans reprezentujący ilość czasu dostępnego w jednej turze). W takim przypadku postać może wylądować w dowolnym miejscu. Żadna technika minimaksowa nie może przeszukać drzewa gry o takim rozmiarze. Zamiast tego stosowane techniki są bardzo podobne do tych stosowanych w grze strategicznej czasu rzeczywistego. Gra turowa często wymaga tego samego rodzaju ruchu postaci AI. W grach turowych rzadko używa się wyrafinowanych algorytmów ruchu. Algorytmy ruchu kinematycznego, a nawet bezpośrednia aktualizacja pozycji (po prostu umieszczenie postaci w odpowiednim miejscu) są w porządku. Na wyższym poziomie planowanie tras, podejmowanie decyzji oraz taktyczna i strategiczna sztuczna inteligencja wykorzystują te same techniki i mają te same szerokie wyzwania. Rysunek przedstawia architekturę AI dla turowych gier strategicznych. Zwróć uwagę na podobieństwo między tą architekturą a architekturą RTS



WYCZUCIE CZASU

Najbardziej oczywistą różnicą między grami strategicznymi opartymi na turach a grami czasu rzeczywistego jest ilość czasu, jaką mają na swoją kolej zarówno komputer, jak i gracz. Biorąc pod uwagę, że nie próbujemy jednocześnie robić ogromnej liczby czasochłonnych czynności (renderowanie, fizyka, tworzenie sieci itp.), istnieje mniejsza potrzeba systemu zarządzania wykonaniem. Często używa się wątków systemu operacyjnego do uruchamiania procesów AI przez kilka sekund. Nie oznacza to jednak, że problemy z czasem nie wchodzą w grę. Gracze mogą normalnie zająć nieograniczoną ilość czasu, aby rozważyć swoje ruchy. Jeśli istnieje duża liczba możliwych jednoczesnych ruchów (takich jak ruchy wojsk, zarządzanie gospodarcze, badania, budowa, i tak dalej), gracz może poświęcić czas na optymalizację kombinacji, aby jak najlepiej wykorzystać turę. Aby konkurować z tym poziomem myślenia stosowanego, sztuczna inteligencja ma ciężką pracę. Część z tego można osiągnąć poprzez projektowanie gier: podejmowanie decyzji dotyczących struktury gry ułatwiającej tworzenie narzędzi AI, wybieranie właściwości fizycznych poziomu łatwych do analizy taktycznej, tworzenie łatwego do przeszukiwania drzewa badań, i używając długości skrętu

które są na tyle małe, że liczba opcji ruchu dla każdej postaci jest możliwa do opanowania. To jednak zaprowadzi cię tylko do tej pory. W końcu potrzebne będzie bardziej znaczące zarządzanie wykonaniem. Podobnie jak w przypadku gry RTS, zazwyczaj istnieje szereg różnych narzędzi decyzyjnych działających w określonych aspektach gry: system ekonomiczny, system badawczy i tak dalej. W grze turowej warto mieć te algorytmy, które szybko zwracają wynik. Jeśli dostępny jest dodatkowy czas, mogą zostać poproszeni o dalsze przetwarzanie. Może to być szczególnie przydatne w przypadku systemu analizy taktycznej, którego obliczenia mogą trwać dłużej.

POMOC GRACZOWI

Inną funkcją sztucznej inteligencji w grach turowych (wykorzystywaną również w niektórych grach RTS, ale w znacznie mniejszym stopniu) jest pomaganie graczom w automatyzacji decyzji, którymi nie chcą się martwić. W Master of Orion 3 gracz mógł przydzielić SI szereg różnych zadań decyzyjnych. Sztuczna inteligencja wykorzystuje następnie tę samą infrastrukturę decyzyjną, której używa w przypadku sił wroga, aby pomóc graczowi. Wspieranie wspomagającej sztucznej inteligencji w ten sposób obejmuje tworzenie narzędzi do podejmowania decyzji, które mają niewielki lub żaden wkład strategiczny z narzędzi decyzyjnych wyższego poziomu. Jeśli na przykład mamy moduł AI do decydowania, na jakiej planecie zbudować kolonię, mógłby podjąć lepszą decyzję, gdyby wiedział, w jakim kierunku strona zamierza najpierw się rozwijać. Bez tej decyzji może wybrać obecnie bezpieczną lokalizację w pobliżu miejsca, w którym prawdopodobnie wybuchnie wojna. Jednak dzięki temu wkładowi z procesu decyzyjnego na wysokim szczeblu, gdy moduł jest używany do wspomagania gracza, musi określić, jaka będzie strategia gracza. Jest to bardzo trudne do zrobienia przez obserwację. Nie znam żadnych gier, które próbowały to zrobić. Master of Orion 3 wykorzystuje bezkontekstowe podejmowanie decyzji, więc ten sam moduł może być używany dla gracza lub strony wroga.