

Streszczenie

Temat: Programowanie sztucznej inteligencji w grach -
System Reguły.

Autor: Piotr Będkowski 19323

Data Opracowania: 18.04.2023

Tworzenie inteligentnych przeciwników w grach komputerowych stanowi wyzwanie dla programistów, którzy starają się, aby zachowanie ich sztucznej inteligencji było jak najbardziej realistyczne i dopasowane do rozgrywki. AI, czyli sztuczna inteligencja, stanowi gałąź informatyki, która skupia się na tworzeniu systemów o zdolności do naśladowania ludzkiego zachowania oraz podejmowania decyzji. W grach wideo AI jest niezbędna do stworzenia przeciwników, sojuszników i postaci nie sterowanych przez gracza. Istnieją różne rodzaje AI, z których każdy ma swoje unikalne cechy, lecz ten dokument poświęcony jest prawdopodobnie najprostszemu z rodzajów sztucznej inteligencji - system regułowy.

System regułowy, w którym AI opiera swoje zachowanie na zestawie reguł, jest jednym z podejść do programowania sztucznej inteligencji. Systemy regułowe są stosunkowo łatwe do zrozumienia i implementacji, dlatego często są wykorzystywane przez początkujących programistów AI. Każda reguła opisuje konkretne działanie AI gdy spełnione są odpowiednie warunki. Przykładowo, jeśli AI zauważy, że posiada niski poziom zdrowia, może zastosować regułę, która każe mu uciekać. W przypadku wysokiego poziomu zdrowia AI może zastosować inną regułę lub kontynuować standardowe działanie. Jednakże, ze względu na swoją sztywność, oznacza to, że systemy regułowe są zazwyczaj zaprojektowane do działania według ściśle określonych zestawów reguł i wytycznych, które zostały wcześniej zdefiniowane, mogą mieć trudności z adaptacją do nowych warunków, które nie zostały uwzględnione w zdefiniowanych regułach, lub z dostosowaniem się do zmieniającego się zachowania przeciwnika czy warunków środowiska. Mimo to, systemy regułowe wciąż mają swoje zastosowanie w programowaniu sztucznej inteligencji i są używane w wielu popularnych grach komputerowych.

Reguła w kontekście sztucznej inteligencji odnosi się do zdefiniowanych instrukcji lub warunków, które są używane do podejmowania decyzji lub wykonywania określonych działań przez system sztucznej inteligencji. Reguły są często zapisywane w postaci "jeśli-wtedy" lub "jeśli-warunek-akcja", gdzie system analizuje warunki i na ich podstawie podejmuje odpowiednie działania. Poniższy przykład prezentuje schemat "jeśli-warunek-akcja": jeśli warunek "zdrowie mniej niż 20" jest spełniony zostanie wykonana akcja "Uciekaj":

```
if(zdrowie < 20) Uciekaj();
```

Reguły w sztucznej inteligencji są stosowane w różnych dziedzinach i służą do modelowania zachowań, podejmowania decyzji, diagnozowania problemów lub udzielania odpowiedzi na zapytania. Reguły mogą być proste i jednoznaczne, ale mogą także być złożone i obejmować wiele warunków, aby uwzględnić bardziej skomplikowane scenariusze.

Przykłady reguł w sztucznej inteligencji obejmują:

1. System rekomendacji filmów:

Reguła 1: Jeśli użytkownik obejrzał trzy filmy tego samego gatunku, zaproponuj mu podobne filmy z tego samego gatunku.

Reguła 2: Jeśli użytkownik ocenił film na wysoką ocenę, zaproponuj mu inne filmy z tego samego reżysera lub z podobnymi aktorami.

W tym przypadku system sztucznej inteligencji analizuje historię użytkownika i stosuje reguły, aby dostarczyć mu spersonalizowane rekomendacje filmów.

2. System zarządzania magazynem:

Reguła 1: Jeśli liczba przedmiotów w magazynie spada poniżej określonego poziomu, zamów więcej towarów.

Reguła 2: Jeśli dane produkty mają krótką datę ważności, umieść je na wyróżnionych półkach.

W tym przypadku system sztucznej inteligencji monitoruje stan magazynu i stosuje reguły, aby utrzymać odpowiedni poziom zapasów i odpowiednio zarządzać produktami.

W przypadku gdy istnieje wiele reguł i może być spełnionych więcej niż jedno warunkowe działanie, można zastosować różne strategie, takie jak:

1) Hierarchia reguł jest strukturą organizacyjną, w której reguły są ułożone w hierarchiczny sposób, aby określić kolejność ich sprawdzania i wykonywania. Hierarchie te polegają na ugrupowaniu reguł na różne poziomy, gdzie reguły na wyższym poziomie mają ogólniejsze warunki, a reguły na niższym poziomie mają bardziej szczegółowe warunki. Przykładowo:

- Jeśli warunek “niskie zdrowie” jest spełniony to zostanie wykonana akcja “Uciekaj”.
- W przeciwnym razie jeśli spełniony jest warunek “gracz w zasięgu” to zostanie wykonana akcja “Atakuj”.
- Jeśli żaden z warunków nie jest spełniony to zostanie wykonana akcja “Patroluj”.

```
if(NiskieZdrowie()) Uciekaj();  
else if (GraczWZasięgu()) Atakuj();  
else Patroluj();
```

2) Wybór losowy: Jeśli kilka reguł jest spełnionych jednocześnie można zastosować losowy wybór, aby AI podjęło jedno z możliwych działań. Na przykład można utworzyć listę potencjalnych akcji, a następnie na podstawie akcji wylosowanej z listy wybierać działanie AI. Przykładowo:

- Jeśli warunek “niskie zdrowie” jest spełniony to do listy zostanie dopisana akcja “Uciekaj”.
- Jeśli warunek “gracz w zasięgu” jest spełniony to do listy zostanie dopisana akcja “Atakuj”.
- Następnie następuje sprawdzenie czy liczba akcji w liście jest większa od zera: jeśli nie to zostanie wykonana akcja “Patroluj”, jeśli tak to zostanie wykonane losowanie, którego wynikiem będzie wylosowanie akcji z listy.

```
if(NiskieZdrowie()) listaAkcji.Dodaj('Uciekaj');  
if(GraczWZasiegu()) listaAkcji.Dodaj('Atakuj');
```

```
if(listaAkcji.Count > 0){  
    r = LosujIndeks();  
    akcja = listaAkcji[r];  
  
    switch(akcja):  
        case 'Ucieczka':  
            Uciekaj();  
            break;  
        case 'Atak':  
            Atakuj();  
            break;  
    } else Patroluj();
```

Kolejnym przykładem jest nadanie akcjom prawdopodobieństwa, które określają szansę na wykonanie odpowiednich akcji. W poniższym przykładzie w zależności od wylosowanej liczby i obecności odpowiednich akcji w liście, podejmowana jest akcja:

- Jeśli wylosowana liczba “r” jest mniejsza lub równa “r_ucieczka” i lista “listaAkcji” zawiera akcję "Ucieczka", wywoływana jest akcja “Uciekaj()”.
- W przeciwnym razie, jeśli wylosowana liczba “r” jest mniejsza lub równa “r_atak” i lista “listaAkcji” zawiera akcję "Atak", wywoływana jest akcja “Atakuj()”.
- Jeśli żadne z powyższych warunków nie jest spełnione, wywoływana jest akcja “Patroluj()”.
- Jeśli lista akcji jest pusta również wywoływana jest akcja “Patroluj()”;

```
double r_ucieczka = 0.3;
double r_atak = 0.9;
if(NiskieZdrowie()) listaAkcji.Dodaj('Ucieczka');
if(GraczWZasiegu()) listaAkcji.Dodaj('Atak');

if(listaAkcji.Count > 0){
    double r = Random.Range(0, 1);

    if(r <= r_ucieczka && listaAkcji.Contains('Ucieczka'))
        Uciekaj();
    else if(r <= r_atak && listaAkcji.Contains('Atak'))
        Atakuj();
    else Patroluj();
} else Patroluj();
```

W zależności od konkretnego przypadku i wymagań, zastosować można jedną z tych strategii lub skombinować je w celu odpowiedniego zarządzania zachowaniem AI. Przykładowo akcje główne mogą być ułożone w strukturze hierarchicznej, a akcje podrzędne wybierane być mogą w sposób losowy:

```
if(NiskieZdrowie()) Uciekaj();
else if (GraczWZasiegu()) Atakuj();
else Patroluj();

void Uciekaj(){
    // losowo wybierz czy AI powinno się schować czy uleczyć
}

void Atakuj(){
    // losowo wybierz formę ataku
}

void Patroluj(){
    // losowo wybierz kierunek patrolowania}
```