Kamil Breczko

Bot do gry MeanMax

25 listopada 2017

Spis treści

1. Opis gry

Gra MeanMax rozgrywa się na okrągłej mapie z 3 graczami. Każdy z graczy jest w posiadaniu 3 pojazdów: *reaper*, *destroyer* i *doof*. W centrum planszy znajduje sie źródło wody. Na mapie istnieją obiekty nie kontrolowane przez grczy, są nimi: *wreck*, *tanker*. Celem tankera jest załadowanie pustego zbiornika wodą, która znajduje się w centrum. Obiekt *tanker* można zniszczyć i w efekcie powstaje *wreck*, z którego *reaper* możne zbierać wode.
Reaper posiada umiejętności:
— zbieranie wody z *wreck*;
— pozostawienie smoły o promieniu 1000, w celu zwiększenia masy pojazdów na podanym polu oraz uniemożliwienia niszczenia *tankerów* (utrzymuje się 3 tury oraz kosztuje 30 jednostek umiejętności) ;
Destroyer posiada umiejętności:
— niszczenia *tankerów*;
— rzucania bomb, które odrzucają przeciwników na odległość 1000 (kosztuje 60 jednostek umiejętności);
Doof posiada umiejętności:
— zwiększenia umiejętności;
— pozostawienie oleju, w celu zwiększenia poślizgu na podanym polu oraz uniemożliwienia zbierania wody (utrzymuje się 3 tury oraz kosztuje 30 jednostek umiejętności);
Gracz wygrywa jeśli uzbiera jak najwięcej wody w ciągu 600 tur lub osiągnie 50 jednostek wody.

2. Opis algorytmu

Rozwiązanie do gry "MenMax"wykorzystuje drzewa decyzyjne. Algorytm należy podzilić na 3 moduły: reaper, destroyer, doof. W których podejmowane są niezależne decyzje. Wszystkie moduły mają wspólny cel, aby wspólnie nazbierać jak najwięcej jednostek wody i utrudniać ruch przeciwnikom. Każdy z pojazdów powinien wykorzysytywać jak najwięcej informacji o położeniu obiektów.
Aby jak najszybciej dotrzeć do podanego celu, pojazdy wykorzystują podaną funkcje regulującą przyśpieszenie:

```
private String getMove(Coordinate coordinate, Coordinate myCoordinate, Coordinate speed) {
    int x = coordinate.getX() - (int) 1.5 * speed.getX();
    int y = coordinate.getY() - (int) 1.5 * speed.getY();
    int throttle = (int) Util.getDistance(coordinate, myCoordinate);
    return x + " " + y + " " + throttle;
}
```

2.1. Reaper

Największą rolą w całej grze odgrywa pojazd *reaper*. Celem procedury zarządzającej pojazdem *reaper* jest wybranie jak najlepszych współrzędnych do zebrania pozostałości wody. W przypadku gdy na mapie nie istnieje rozbity *tanker*, pojazd wykonuje ruch do *destroyer*, odpowiadającemu za niszczenie tankerów. Jeśli plam wody jest więcej niż jedna, algorytm wykorzystuje funkcje oceniającą każdy *wreck* z osobna. Ocena danego wrecka wykorzystuje takie informacje jak: kierunek, ilość jednostek wody z wreców znajdujących się w pobliżu, ilość pojazdów znajdujących się w pobliżu, promień obiektu, położenie względem centrum planszy oraz kierunek, w którym znajduje sie względem *reaper'a*. Wszystkie wartości są przeskalowane od 0 do 100, z precyzją do 10 miejsc po przecinku oraz przemnożone przez odpowiedni koszt.

```
rateDistance = - 1/120;
rateRadius = 1/10 * 0.01;
rateExtra = 100/8 * 0.3;
rateDirection = (isSameDirection(reaperCoordinate, coordinate)) ? 1.0 : 0.9;
rateCollection = 0.8;
rateCenter = - 1/60 * 0.01;
ratePopularity = -2;
```

Funkcja największy nacisk kładzie na odległość, pomiędzy *wreck* a *reaper'em*. Im większa odległość tym ocena jest mniejsza. Na spadek oceny ma wpływ ilość pojazdów znajdujących się w pobliżu. Pojazd *tanker* liczy się jako 2 pojazdy, ze względu na wielkość. Z obserwacji wyników testów, w przypadku napotknia tankera na ścieżce, reaper jest zablokowany na kilka tur. Na ocene wpływ ma także kierunek w jakim *reaper* musi się poruszyć, ponieważ jeśli *wreck* jest z tyłu to przyśpieszenie zeruje się, co spowalnia i martuje turę. Ocena jest zwiększana jeżeli w pobliżu znajdują się inne wracki. W przypadku gdy algorytm wybierze *wreck*, na którym nakłada się inny *wreck* to wybierana jest najlepsza współrzędna (punkt środkowy), w celu zebrania podwójnej ilości jednostek wody.

2.2. Destroyer

Celem procedury zarządzającej pojazdem *destroyer* jest wybranie najlepszego w danym momencie tankera, który po zniszczeniu zwróci jednostki wody do zebrania. W przypadku gdy na mapie nie istnieje *tanker*, pojazd nic nie robi. Z obserwacji wyników testów, przypadków gdy nie ma *tankerów* na mapie jest mało. W przypadku gdy *tankerów* jest więcej, algorytm wykorzystuje funkcje oceniającą każdy z *tankerów*. Funkcja oceny danego tankera wykorzystuje te same informacje co funkcja oceniająca *reaper'a* i dodatkowo: pojemność danego *tankera*

oraz ilość wolnego miejsca na jednostki wody. Wszystkie wartości są przeskalowane od 0 do 100, z precyzją do 10 miejsc po przecinku oraz przemnożone przez odpowiedni koszt.

```
rateTankersNumber = 2;
rateDistance = - 1/120;
rateRadius = 1/10 * 0.01;
rateExtra = 100/8 * 0.4;
rateDirection = (isSameDirection(destroyerCoordinate, coordinate)) ? 1.0 : 0.8;
rateCollection = 1.1;
rateCenter = - 1/60 * 0.01;
ratePopularity = -2;
rateEmptySpace = - 100/extra2 * 0.6;
```

Podobnie jak przy ocenie *wrecku* na spadek oceny danego tankera wpływa ilość wrogich pojazdów znajdująca sie w pobliżu oraz odległość. Większe znaczenie ma kierunek ruchu do którego *destroyer* musi się obrócić, ponieważ pojazd wolniej rozpędza się ze względu na swoją mase. Ocena zwiększa się, jeśli w pobliżu znajdują się inne *tankery* oraz plamy wody.
Algorytm używa także dodatkowe funkcjonalności pojazdu, a mianowicie rzucanie granatów. Granat rzucany jest na współrzędne, na których znajduje się nasz reaper. Gdy na współrzędnych znajduje się pojazd to efekt odczują tylko te pojazdy, które są w pobliżu. Aby rzucić granatem należy spełniać kilka warunków:
— wystarczająca ilość umiejętności;
— *destroyer* musi być w pobliżu *reaper'a*;
— jeśli od ostatniego rzucenia granatu upłyneły 5 tur;

2.3. Doof

Celem procedury zarządzającej pojazdem *doof* jest zbieranie umiejętności oraz utudnianie poruszanie się przeciwnikom. Algorytm wybiera z pośród 3 graczy jednego, który ma aktualnie największa ilość jednostek wody i kieruje się na współrzędne na których znajduje sie reaper. Jeśli pojazd *doof* znajduje się blisko wrogiego *reaper'a* oraz spełnia podane poniżej warunki to rozlewa plamę oleju.

Warunki, które musi spełniać pojazd *doof* aby rozlać olej:
— wystarczająca ilość umiejętności;
— w pobliżu wrogiego *reaper'a* nie znajduje się nasz reaper;
— jeśli od ostatniego rozlania oleju upłyneły 3 tury;