Aplicatie joc:

Pike and shot

Grupa 30234

Biz Adrian si Sebastian Alexandru Sabou

Contents

1.Enuntul Problemei

1.1 Enuntul

Sa se implementeze un joc care va reprezenta o problema ce poate fi rezolvata folosind 3 algoritmi de cautare in spatiul starilor neinformat si unul informat.

1.2 Aplicatia

1.2.1 Prezentare problema

Pe o harta uni dimensionala exista 10 (0-9) poziti reprezentand pozitile de asezare a 2 armate pe campul de batalie. Linia frontului este mereu la mijloc intre pozitia 4 si 5. Armatele sunt alcatuite din maxim 3 tipuri de unitati: muschetari, halbardier si cavalerie. Scopul este ca o armata sa o invinga pe cealalta. Pe campul de batalie se pot face 2 tipuri de actiuni: atac sau mutare. Toate unitatile au o limita de actiuni pe care le pot lua pe tura. O data cu terminarea turei se reseteaza punctele de actiune. Unitatiile vor sta mereu cat mai aproape de linia frontului (mijlocul hartii).

Atac este atunci cand o unitate loveste una inamica cauzand damage, pierzand un punct de actiune.

Mutare este atunci cand doua unitati din aceeasi parte isi schimba locurile intre ele, ambele suferind un punct de actiune.

Ca dezvoltare, pentru a creste gradul de dificultate se poate crea un oponent activ.

1.2.2 Unitatile

Muschetarii: au o stamina de 10 hp(healt points), cu 2 ap(action points). Acestia pot ataca o unitate inamica care se afla direct in fata lor sau la o unitate distanta. Ei cauzeaza daune de 5 puncte contra muschetarilor si halbadierilor, dar numai 3 contra cavaleriei.

Halbardier: au o stamina de 10 hp(healt points), cu 2 ap(action points). Acestia pot ataca o unitate inamica care se afla direct in fata lor. Ei cauzeaza daune de 3 puncte contra muschetarilor, 5 contra altor halbardieri si 7 contra cavaleriei.

Cavaleria: au o stamina de 10 hp(healt points), cu 3 ap(action points). Acestia pot ataca o unitate inamica care se afla direct in fata lor. Ei cauzeaza daune de 7 puncte contra muschetarilor, 3 contra altor halbardieri si 5 contra cavaleriei.

2. Modelarea problemei

Reprezentarea campului se face folosind un vector cu 10 pozitii. Primele 5 apartinand primei tabere, iar celelalte 5 taberei opuse. Vectorul va contine instante a unei clase numite “Unit”.

Pentru a reprezenta unitatiilor se vor folosi literele M pentru muschetarii, H pentru halbardier si C pentru cavalerie.

Actiunile posibile fara a lua in calcul tipul si pozitia unitatilor sunt:

Atac, mutare(intre unitati, swich) si next-turn.

Pentru starea de inceput vectorul trebuie sa aiba unitati in cel putin una sin primele 5 si cel putin una in ultimele 5 pozitii.

Pentru starea de final vectorul trebuie sa aiba cel putin o unitate in primele 5 pozitii si ultimele 5 pozitii sa fie goale.

3.Algoritmii folositi

Chiar daca algoritmii sunt meniti sa ofere o cautare neiformata si neasistata in mare masura, niste limite au fost necesar impuse pentru evitarea cazurilor de intrare in situatii de bucla continua. O astfel de limitare este declararea unei stari invalide daca aceasta are mai mult de 100 de ture date.

3.1 DFS

Cautarea in adancime creaza toate posibilitatile ce pot deriva dintr-o stare curenta si incepe parcurgerea uneia pana ce ajunge la situatie de “frunza”(nu poate genera noi stari), dupa care va continua cu urmatoarea stare generata la punctul anterior.

Acesta se va opri cand va gasi o stare de goal(rezultat, victorie) sau pana ce va fi parcurs toate starile. Adeseori ajunge la o solutie chiar si daca nu e cea optimala.

3.2 BFS

Cautarea in latime creaza toate posibilitatile ce pot deriva dintr-o stare curenta si incepe parcurgerea uneia generand la nivelul ei toate posibilitatile, dupa care va continua cu urmatoarea stare generata la punctul anterior. Acesta se va opri cand va gasi o stare de goal(rezultat, victorie) sau pana ce va fi parcurs toate starile. Aceasta va ajunge la solutia optimala dar in timp foarte lung.

3.3 Random Search

Aceasta cautare alege aleatoriu viitoarea stare pe care o va genera si o va accesa.

Neavand nicio logica sau planificare este posibil ca aceasta sa nu gaseasca o solutie in timplul acordat.

3.4 Uniform Cost Search

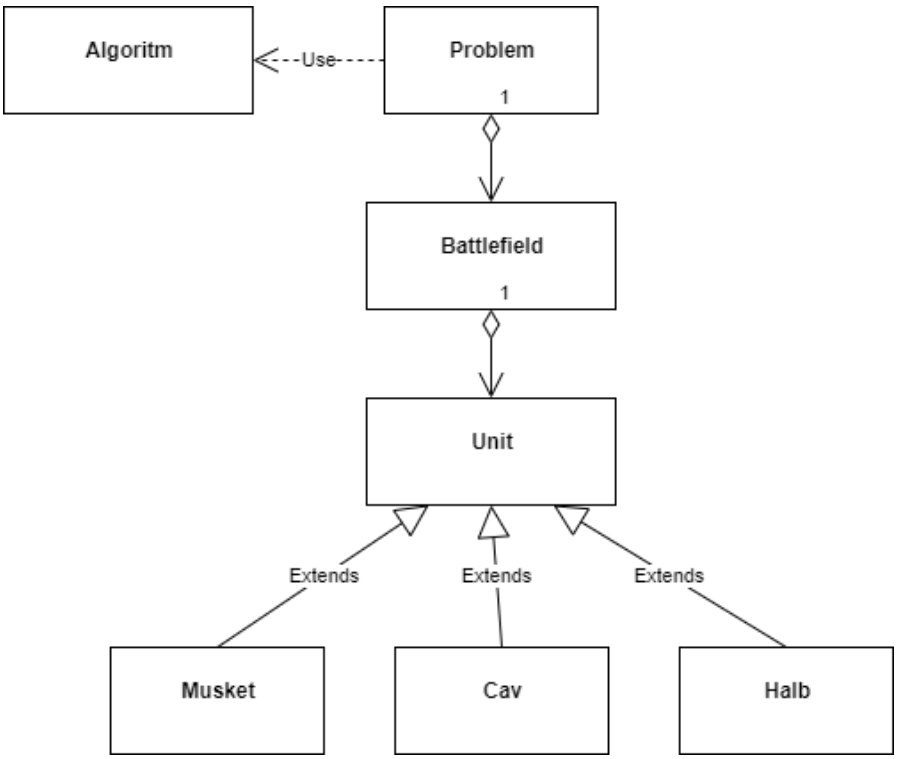
Aceasta cautare genereaza toate posibilitatile pornind de la o stare data atasand fiecareia un cost de parcurgere. Urmatoarea stare pe care o va lua fiind cea cu costul cel mai mic. Urmand din nou sa genereze toate posibilitatile si sa aleaga din multimea totala de posibilitati generate o noua stare.

3.5 A*

Acest algoritm este unul informat, dar se aseamana foarte mult cu cel de mai sus. El diferentindu-se de cel de costuri uniforme printr-o functie euristica ce aproximeaza “optimist” costul de a ajunge la rezultat.

4.Implementare

4.1Clasele



Battlefield este clasa ce memoreaza unitatile in forma de vector uni-dimensional si cuprinde casele de interactiune(actiunile) dintre unitati si de next turn.

In Unit se afla metodele de schimbare a statusurilor unitatilor. Clasele ce extind Unit reprezinta tipurile de unitati acestea suprascriu metoda de damage contra altor unitati.

Clasa Problem cuprinde metode ce ofera informatii despre stari si genereaza urmatoarele posibile stari.

In fisierul Algoritm este definita clasa nod si algoritmii de cautare in spatiul starilor.

Functiile heuristice pentru algoritmul informat constau in aproximarea in functie de cate unitati mai sunt in battlefield de partea inamica si ce tip sunt in comparatie cu cele din echipa jucatorului.

4.2Rularea in consola

Aplicatia poate fi rulata in interfata consola executand fisierul Game.py.

Acesta va oferi un battlefield aleatoriu sau bazat pe un sablon.

Pentru a face o actiune se va da pozitia untiati care va face actiunea si pozitia unitatii cu care va interactiona.

Daca e o unitate vecina aliata va face swap sau daca e o unitate inamica va ataca(scazand puncte de viata de la unitatea inamica dar si de la sine prin mecanica de contra atack). Fiecare actiune va consuma puncte de actiune(AP) de la aceea unitate si daca e cazul de swap si de la unitatea cu care interactioneaza

Se poate da tura oferind pozitia -1, aceasta va incrementa contorul de tura si va reseta punctele de actiune ale unitatiilor.

5.Rezultate

	DFS	BFS	UCS	A*A	A*T
easy	Neoptimal, 505 de noduri, 0.0855 seconds	Optimal, 923 de noduri, 0.2173 seconds	Optimal, 91 de noduri, 0.1762 seconds	Optimal, 78 de noduri, 0.1388 seconds	Optimal, 89 de noduri, 0.1762 seconds
mediu	Neoptimal, 25 de noduri, 0.0042 seconds	Optimal, 3680 de noduri, 1.7984 seconds	Neoptimal, 192 de noduri, 0.5545 seconds	Neoptimal, 156 de noduri, 0.4491 seconds	Neoptimal, 580 de noduri, 2.0627 seconds
hard			Neoptimal, 2912 de noduri, 27.6295 seconds	Neoptimal, 759 de noduri, 6.4890 seconds	Neoptimal, 2599 de noduri, 23.7674 seconds