Pathfinding

1 Krav

Algoritmen behöver nog egentligen inte vara så snabb, eftersom monster inte kommer ha så lång aggrorange, kanske en skärm eller nått. Å andra sidan så kan den behöva köras ofta, om målet flyttar på sig. Några önskvärda egenskaper:

- Om man kan så borde man gå raka vägen
- Det vore bra om algoritmen kunde hantera mobs av olika storlek

Vanliga tilebaserade algoritmer har inte dessa egenskaper. Det är nog dock ganska lätt att göra om en path till en rak väg med lite post-processing.

Mobs av olika storlek är nog svårare. Det är främst bossmobs, t.ex. drakar o.s.v. som kanske man vill ska vara stora. Säg att man vill kunna ha 2x2-mobs. En grej man skulle kunna göra är att tänka på det när man gör kartan. Drakar kanske bara finns nere i en viss grotta, så man skulle kunna göra terrängen extra lämplig för 2x2-pathfinding just där. Ett annat alternativ är att bara skita i det, och låta drakar se stora ut men ändå vara 1x1 i pathfindingen. Lite "buggigt" kan det vara utan att man stör sig. EQ är ett exempel på det.

En annan grej att ha i åtanke är att det är inte viktigt att hitta kortaste vägen om pathfindingen inte används för spelare. Ingen kommer att bry sig om att en mob tar en omväg ibland. Detta gör att man kan göra pathfindingen snabbare, genom att använda en heuristik som är praktisk fast kanske inte alltid admissible.

2 A* med tiles

A* är lätt att implementera, och funkar bra, förutom för större mobs.

3 Triangulation A*

Det finns 2 artiklar som beskriver denna algoritm, en på typ 15 sidor och en på över 100. Så det finns i alla fall beskrivet hur man ska göra.

Algoritmen går ut på att dela upp kartan i trianglar, och sedan använda trianglarna som noder i en graf, och köra A* på den. När man hittat en stig av

trianglar så räknar man fram den faktiska kortarste vägen genom den. Sedan fortsätter man att leta efter flera stigar. Man slutar när en optimistisk (admissible) uppskattning av vägen genom nästa stig av trianglar man hittar är längre än den kortaste faktiska vägen man hittat.

3.1 Uppdelning i trianglar

Man kan inte dela upp kartan hur som helst. T.ex. skulle det inte gå att hitta de största rektanglarna och dela dem till trianglar. En kant av en triangel måste gränsa till en annan triangel eller ett hinder. En sida kan inte gränsa till fler än en triangle eller ett hinder. Jag vet inte hur komplicerad algoritmen är.

3.2 Heuristik

Låt oss kalla den upskattade kvarvarande vägen från en triangel till målet för h och den upskattade vägen från startpunkten till den triangeln man står i (i sökningen) för g. En triangels ingångskant är kanten på triangels som man gick över för att komma in i triangeln. Man kan skatta h med raka vägen mellan målet och ingångskantens närmsta punkt till målet. g kan skattas på olika sätt:

- Kortaste vägen mellan start och ingångskanten
- Avståndet mellan start och mål minus h för noden
- Förälderns g plus förälderns h minus nodens h
- Förälderns g plus avståndet mellan förälderns ingångskant och nodens ingångskant

(Jag förstår alla förutom den sista.)

Alla dessa är admissible, så man kan ta det värde som är störst av dem som g. Ett stort g är bra.

3.3 Faktiska vägen

När man har hittat en stig av noder (trianglar) så använder man nått som heter ”funneling algorithm”. Jag vet inte hur komplicerad den är.

3.4 Olika stora mobs

Det ingår också att räkna ut, för varje triangel, hur stora mobs som kan gå igenom dem (olika beroende på vilken sida man går till och från). Det verkar lite jobbigt.

4 Slutsats

TA* ska vara snabb, och dessutom är den flexibel och klarar våra krav. Dock så kan den nog vara jobbig att implementera. Dessutom så är A* en del av den, så man förlorar inget på att göra A* först.