UMEÅ UNIVERSITET
Institutionen för Datavetenskap
Rapport obligatorisk uppgift

Applikationsutveckling i Java, 7.5 hp 5DV135

Bilaga: AntiTD Artificiell Intelligens & Verktyg

Obligatorisk uppgift nr

2

Namn	Alexander Beliaev	**albe0060**	**c15abv**
Namn	Alexander Ekström	**alek0013**	**c15aem**
Namn	Jan Nylén	**jany0014**	**id12jnn**
Namn	Karolina Jonzén	**kajo0136**	**id12kjn**
Grupp	9		
Datum	19 december, 2016		
Kursansvarig	Johan Eliasson		
Övriga lärare	Jan Erik Moström		
Handledare	Adam Dahlgren Lindström, Alexander Sutherland, Filip Allberg,		
Källkod	labb: /home/c15/c15abv/edu/javaApp/labb2_grupp9 denna bilaga: /home/c15/c15abv/edu/javaApp/labb2_grupp9/antitd_bilagor		

Kommentarer:

Innehållsförteckning

1 Inledning

Detta är en bilaga till den ordinarie rapport som är förknippad till projektet. Vi går här in lite djupare i vilka de olika "intelligenta" komponenter i systemet som aktivt får spelet att gå framåt är, samt hur de bär sig åt. Vi kollar även på ett verktyg framtaget för att träna upp den artificiella intelligens som används av motspelaren.

2 Översikt

Verktygen som skapats återfinns i den exekverbara .jar filen aitrainer.jar. De träningssessioner som körs sker i riktigt tid, just precis som om man skulle spela spelet normalt. Nedan är filer förknippade med denna bilaga:

```
antitd_bilagor/
├── aitrainer.jar
├── memory
├── Levels.xml
└── antitd_grupp9_aiochverktyg.pdf
```

3 Användarhandledning

För att använda verktyget krävs det först och främst att användaren innehar en fil innehållandes en specifikation över olika nivåer. Denna fil ska vara i .xml format och använda sig av Jan Nyléns konvention över nivåer (se ordinarie rapport). Den andra filen som krävs är en vanlig fil; har man aldrig kört verktyget förrut räcker det med en tom fil. Denna fil kommer successivt att fyllas på med data, och bör därför ej modifieras. Nedan följer exempel på körning.

Listing 1: Argument vid körning

```
java -jar aitrainer.jar [level file] [level] [memory file]
   [iterations]
```

Listing 2: Exempelkörning av verktyget med 100 iterationer

```
java -jar aitrainer.jar Levels.xml 2 memory 100
```

[level file] samt [memory file] behandlas som strängar medan [level] och [iterations] som heltal. I ovanstående exempel finns filerna under samma sökväg som verktyget. Inga robusta felkontroller görs av användarinput.

4 AI för varelser

AI:n som används till varelsernas förflyttning i spelet använder sig främst av en djuppet först sökning. Den är dock ganska modifierad i den bemärkelsen att man först kollar på figurens riktning, gör ett val beroende på vilka rikningar den redan besökt utifrån den nuvarande positionen, och sedan fortsätter sökningen. Om figuren dessutom av någon anledning (exempelvis vid utplacering av många teleportrar) kan figuren komma att traversera samma position flera gånger. För att undvika att AI:n till figuren tror att figuren redan besökt alla positioner på hela kartan, slängs istället figurens nuvarande minne bort, och sökningen fortsätter.

5 AI för torn

Torens AI är troligen den minst sofistikerade i spelet eftersom man här endast gör enkla kontroller och jämförelser. Om ett torn inte redan har ett mål, eller om dess mål är utom räckvidd eller dött, kommer AI:n att göra en ny sökning av attackerarens figurer med tornets räckvidd som parameter. Alla figurer som den hittar jämförs därefter med varandra där prioritetsordningen i grova drag är figurens form, färg och därpå kvarvarande antal liv. Ett torn som redan har ett mål låser fast på det målet tills ovannämnda kriterie uppfylls.

6 AI för motspelaren

Motspelarens AI är inspirerad av genetisk evolution och mutation. Däremot består AI:n endast av ett lager och använder därför heller inget neurologiskt nätverk. Vad den i grund och botten faktiskt gör är att den slumpmässigt placerar ut torn. Om en viss mängd torn som placerats ut på specifika platser inom en viss tid resulterar i att den lyckas väldigt bra med att förhindra att spelaren klarar en karta kommer den spara dessa värden. När den nästa gång ska köra en liknande eller samma bana kommer AI:n att hämta det specifika minnet och försöka utföra det den gjorde den gången då det gick väldigt bra. Här kan man däremot även ändra så att det finns en förhöjd (eller förminskad) chans att AI:n avviker från sitt minne/plan gällande något eller några torn. Med andra ord om man specificerar att det finns en chans att ett torn kan läggas ut med ett deltavärde - låt säga 500 millisekunder - och tornet egentligen ska läggas ut vid tiden 3345 millisekunder, finns det en möjlighet att tornet läggs ut när som helst på intervallet $[2845, 3845]$.