Artificiell intelligens för digitala spel DA308A: Inlämningsuppgift #2

Färdig 11 mars 2016

13:15

Uppgift 1

Spela spelen/använd programvaran:

- Galactic Arms Race (GAR),
- PicBreeder,
- Dance Evolution och
- NERO Game samt
- Petalz.

(Länkar finns på *It's Learning* i mappen "Inlämningsuppgift 2 – Resurser").
Läs artikeln

- "Evolving Content in the Galactic Arms Race Video Game" (Hastings et Al, 2009).

Uppgift 2

Den här uppgiften består av **två** moment – en rapport och en programmeringsuppgift. Ha båda tillgängliga när du redovisar ditt arbete.
Programmeringsuppgiften innebär att du ska implementera en genetisk algoritm (**GA**) eller annan form av evolutionär beräkning (**EC**). GA:n skall lösa ett eget valt spelproblem. Notera att problemet inte får vara av trivial natur (se kurslitteraturen s 513-521, specifik s 518-521). **Om ditt problem är trivialt kommer det inte att godkännas** så kontrollera med lärare innan du påbörjar uppgiften. Pseudokod **??** visar hur principen för genetiska algoritmer fungerar.
Rapporten ska vara relativt kortfattad men den måste redovisa följande;

- **problemet**,
- **representationen** (hur representeras lösningen i din GA? Ex.: om du använder en heltalsvektor för att representera lösningen, vad betyder de olika fälten och vad betyder olika värden eller olika intervall),
- hur "**fitness-funktionen**" fungerar,
- **evolutionära parametrar** (strategi, selektering, mutation, etc.)
- och **resultatet** för bästa, sämsta, median, medel, standardavvikelse för "fitness"-funktionen för generationerna 1, 10, 100 och 1000 (lämpligen med en tabell).
- samt visa resultaten i en graf.

Pröva att göra någon körning mot 10000 generationer och se *om* och *kring* vilken generation du hittar punkten där förbättringen i fitnessvärdet planar ut. En sådan punkt kan vara en punkt där GA:n ska sluta köras.

Listing 1: Exempel på genetisk algoritm (GA)

```
// Genetic Algorithm
#include <stdio.h>
#define GENERATIONS 1000

float chanceValue = 0.1f;
int numberOfMembers = 100;
int population[numberOfMembers];

main(){
     initializePopulation(population());
     for (int counter = 0; counter < GENERATIONS; counter++){
          fitnessFunction(population[]);
          sortAccordingToFitnessValue(population[]);
          BreedTopPopulation(population[]);
          mutateHalfOfPopulation(population[], chanceValue);
     }
     fitnessFunction(population[]);
     sortAccordingToFitnessValue(population[]);
     printf("Best solution " + population[0]);
}
```

Uppgift 3

Denna uppgift är främst orienterande natur. Syftet är att du ska bekanta dig med artificiella neuronnät (**ANN** - Artificial neural network). I zip-filen "ANN-demo.zip" på *It's Learning* i mappen "Inlämningsuppgift 2 – Resurser/ANN Demo" finns ett demonstrationsprojekt för ANN.
Intressant att titta närmare på är filen "MyNeuralNetwork.cpp". I klassen "NeuralNetworkLayer" finns en implementering av ett fler-lager feed-forward nät. Den implementeringen hanterar minnesallokering och vikter i ANN:et (se "MyNeuralNetwork.h" för metodnamn). De olika lagren är kopplade som förälder-barn-relationer. Du bör titta på konstruktorn och initieringsmetoden för att få en översikt över hur ett ANN kan konfigureras.

```
NeuralNetworkLayer::NeuralNetworkLayer()
void NeuralNetworkLayer::Initialize(int NumNodes, NeuralNetworkLayer* parent,
NeuralNetworkLayer* child)
```

Programmet är en demonstration av "flocking" och "chasing" beteenden där de AI-kontrollerade enheterna mha ett ANN beslutar om de ska CHASE (jaga spelaren), EVADE (fly från spelaren) eller FLOCK (ansluta till andra AI-kontrollerade enheter). Demonstrationen är ett enkelt spelscenario där ett antal enheter kan anfalla spelaren. Istället för att fullt ut använda en FSM så använder vi ett ANN som kan förändra sitt beteende beroende på hur spelaren agerar.
Ungefär 20 AI-styrda enheter rör sig runt på skärmen och kan anfalla, fly eller gruppera. Du kan flytta spelaren. En strid mellan en AI-kontrollerad enhet och spelaren hanteras genom att spelaren och AI-kontrollerade enheter förlorar livspoäng när de är i närheten av varandra. Spelaren förlorar mer livspoäng om flera AI-kontrollerade enheter är inom stridsavståndet. När en enhet förlorat alla sina poäng dör en och den dyker upp automatiskt. Under spelets gång användes ANN som beslutsfattare och tränas under tiden (mha back-propagation algoritmen) och de AI-kontrollerade enheterna utvecklas kollektivt.
Din uppgift är att ta reda på vilket beteende de AI-kontrollerade enheterna lär sig (det finns två möjligheter) genom att köra demonstrationen ett antal gånger. Använd 1-8 för att ange vilken mängd skada som utdelas (1 betyder lite och 8 betyder extremt mycket).