Gaming AI

Projekt Big Data

Friedrich Braun, Valentin Krön
Betreuer: Eugen Betke, Julian Kunkel

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2017-03-19

Gliederung (Agenda)

Altes Projektziel

- AI-Bot für RTS
  - genauer: Spring-Engine
- Training via Genetischem Algorithmus
- Sprache C oder C++
- Orientierung an bestehenden AI's
  - gegen bestehende AI's spielen
- AI gleichmächtig wie Mensch

  → besserer Spieler

Warum Änderung des Ziels?

- Spring ist unübersichtlich aufgebaut
  - nicht nachvollziehbare Probleme beim Starten von Spring
  - nicht modspezifische AI's ließen sich nicht einbinden
- Mods mit eigener Lobby
  - funktioniert in sich gut
  - nicht von außen zu erweitern
  - Mod um AI erweitern (Lua) zu aufwendig
- Ziele nicht im Projektzeitraum erreichbar

Neues Projektziel

- Proof-of-concept
  - AI für RTS
  - Training mit genetischem Algorithmus

$\Rightarrow$ Vereinfachtes RTS selber bauen

  - Spieleengine Unity [1]
- Beibehaltung des direkten Vergleiches mit Mensch

Warum Unity?

- Sollte Spiel bleiben
- AI weiterhin gleichmächtig wie Mensch
- bekannt
  - umfassende Dokumentation
  - viele (kostenlose) Online-Tutorien
- kostenlos für Uniprojekte

Schwierigkeiten mit Unity

- Unity handhabt Codestrukturen anders
  - $\Rightarrow$ viele Konventionen nicht anwendbar
  - $\Rightarrow$ unintuitiv
- Multiplayer (Unity Networking)
  - jeder Wert muss explizit übertragen werden
  - Fehler durch verzögerte Übertragung
  - Objekt-Referenzen nicht übertragbar
  - netId-Component für uns nicht zugreifbar

Warum C# ? Warum Visual Studio?

- C#
  - Aktuelle Unityversion unterstüzt C# und Javascript
  - bewusst gegen Javascript und für C# entschieden
  - C# relativ nah an bisher Gelerntem
- Visual Studio
  - direkte Verbindung mit Unity
  - unkompliziert und einfach
  - keine Suche nach anderer IDE

Neuronales Netzwerk

- Warum Verwendung eines NN
  - Schwarmintelligenz
    - viele Teilprobleme geringer Koplexität
    - Teilprobleme können dieselbe Lösung haben
  - skalierbare Komplexität
  - Erweiterbarkeit um neue Funktionalitäten
  - generieren von Entscheidungen
- Umsetzung
  - Bibliothek eingebunden
    - C# Neural Network Library [2]
    - nur Übernahme der Struktur

Aufbau

- selbstgebaut in C#
- Gene sind die Gewichte des NN
- Genotyp = Phänotyp
- Fitnesswert
  - eigene Einheiten - Gegnereinheiten
- zufällige Initialisierung

Kennfunktionen

- Rekombination
  - random cut
- Mutation
  - pro Individuum ob mutiert
  - wenn, dann genau ein Gen zufällig neu
- Selektion
  - die Fittesten überleben

Was erreicht?

- Spiel im lokalen Netzwerk voll funktionsfähig
  - genau zwei Spieler
- Gleichmächtigkeit erfüllt
- unterschiedliche Individuen erzeugen unterschiedliche Verhaltensweisen
- der genetische Algorithmus funktioniert
  - Kommunikation von Gewichten und Fitness funktioniert

Ausblick

- Spiel startet nicht vollautomatisch
  - $\Rightarrow$ Training nicht vollautomatisch
  - $\Rightarrow$ keine Trainingserfolge
- Training parallelisieren
  - Unity kann nicht im Background laufen
- Erweiterung der Komplexität des Spieles

Quellen

1. https://unity3d.com/
2. http://franck.fleurey.free.fr/NeuralNetwork/
3. https://msdn.microsoft.com/library/