Ohjelmistokehityksen teknologioita - Seminaarityö

Minimax-algoritmilla ristinollaa pelaava botti

Tietorakenteet ja algoritmit

Arina Soldan

14.4.2022

Sisältö

Tiivistelmä

Minun tavoitteeni oli oppia yksinkertaisten algoritmien soveltamista pelikehityksessä, ja tässä yhteydessä tutustuin minimax-algoritmiin. Valitsin Unityn kehitysalustaksi, koska minulla oli aiempaa kokemusta pelien luomisesta siellä. Päätin kehittää ristinollapelin, jossa pelaaja ei voisi voittaa vastustajaa, joka toimii bottina minimax-algoritmin avulla.

Käytin C#-kieltä pelin toteuttamiseen ja hyödynsin ChatGPT:tä apuna koodin kirjoittamisessa. Aloin tutkia minimax-algoritmia ja sen soveltamista pelitekoälyn luomiseen. Kehitin pelilogiikan, joka mahdollistaa pelaajan ja botti-vastustajan vuorottelevan siirrot, mutta botti oli optimoitu siten, että se valitsi aina parhaan mahdollisen siirron minimax-algoritmin avulla.

Tärkein tulos oli onnistunut ristinollapelin toteutus, jossa botti oli älykäs ja käytti minimax-algoritmia estääkseen pelaajaa voittamasta. Projektin myötä sain syvällisempää ymmärrystä algoritmeista pelikehityksessä ja olin tyytyväinen siihen, kuinka ChatGPT helpotti koodin kirjoittamista ja kehitysprosessia.

1 Johdanto

Ristinolla on klassinen yksinkertainen monille tuttu strategiapeli.

Tämä projekti keskittyy ristinollapelin toteuttamiseen Unity-työkalulla käyttäen C#-ohjelmointikieltä sekä kuinka integroida minimax-algoritmi.

Tutkimukseni tavoitteena on paitsi luoda toimiva ristinollapeli myös tarjota oppimiskokemus pelinkehityksen perusteista, interaktiivisen käyttöliittymän toteuttamisesta ja tekoälyn integroinnista peliprojektiin.

1.1 Työvaiheet

1. **Unity-projektin luonti:**

Unity-projektin luonti, asetuksien määrääminen ja kuvien valinta ristille, nollalle ja pelikentälle.

2. **Pelikentän luominen:**

Pelikentän, ristien ja nollien luonti Unityn GameObjecteina.

3. **Pelilogiikan luominen C#:llä:**

C#-skriptin luominen käyttäen pelilogiikan hallitsemiseksi. Määritteleminen pelaajien vuorottainen toiminta ja logiikka sille, miten peli etenee jokaisen siirron jälkeen.

4. **Minimax-algoritmin toteuttaminen**

Toteutaminen minimax-algoritmin C#-skriptillä pelin tekoälyn käyttämiseksi sekä integroiminen pelin logiikkaan siten, että tekoäly voi tehdä älykkäitä päätöksiä pelaajaa vastaan.

2 *Käytetyt tekniikat*

2.1 Pelin ja yleisen logiikan luonti

Aloitin toteuttamalla pelilaudan Unitin objektina.

Sijoitin siihen 9 painiketta, jotka vaihtuvat pelin aikana. Kun pelaaja painaa yhtä nappia omassa vuorossa, nappiin ilmestyy Sprite eli kuvake, joka kuvaa nolla, kun taas botti painaa omassa vuorossa nappia, se muuttuu ristin-Spriteksi.

Kuva 1. Objektit valmiina

Kuva 2. Käytetyt Spritit

Kumpikin nappi tulee epäaktiiviseksi jokaisen vuoron jälkeen, joten niitä ei voi painaa uudelleen.

Sitten lisäsin yksinkertaisen vuorojen logiikan - jos on pelaajan vuoro, se on totta, jos botin, se on epätosi.

Lisäsin napit 3x3 matriisiin. Valitettavasti Unity ei automaattisesti tunnista kaikki napit GameObjectista, joten lisäsin jokaisen napin erikseen koodiin.(Kuva 3)

```
// Liitetään jokainen nappi pelilaudalle jokaiseen soluun
buttons[0, 0] = GameObject.Find("Button1").GetComponent<Button>();
buttons[0, 1] = GameObject.Find("Button2").GetComponent<Button>();
buttons[0, 2] = GameObject.Find("Button3").GetComponent<Button>();

buttons[1, 0] = GameObject.Find("Button4").GetComponent<Button>();
buttons[1, 1] = GameObject.Find("Button5").GetComponent<Button>();
buttons[1, 2] = GameObject.Find("Button6").GetComponent<Button>();

buttons[2, 0] = GameObject.Find("Button7").GetComponent<Button>();
buttons[2, 1] = GameObject.Find("Button8").GetComponent<Button>();
buttons[2, 2] = GameObject.Find("Button9").GetComponent<Button>();
```

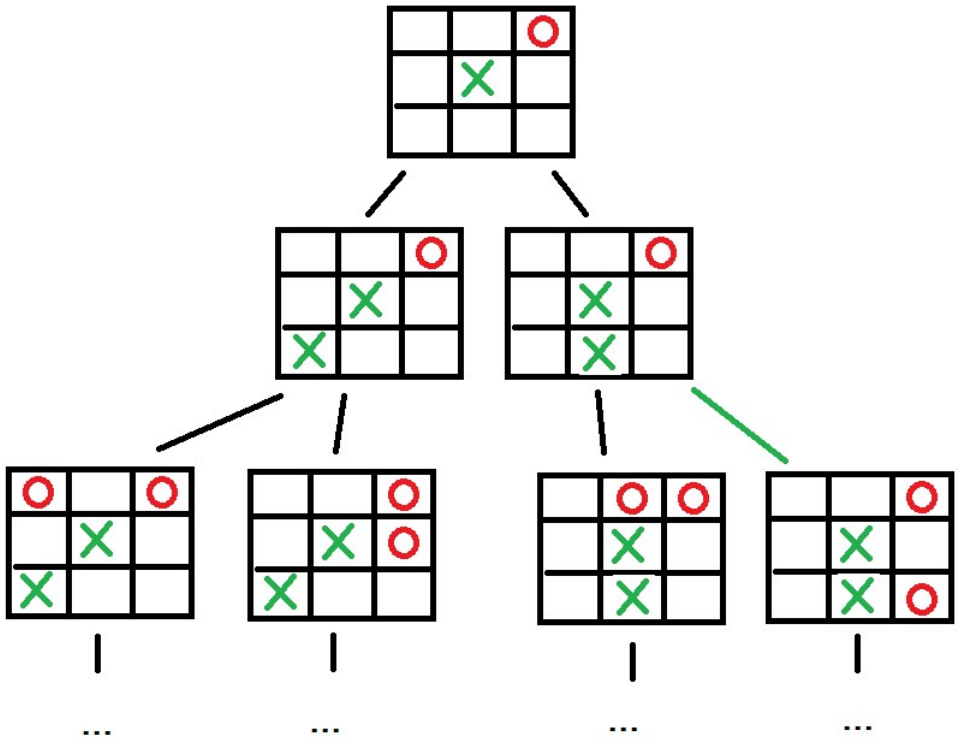
Kuva 3. Matriisin luonti

Nyt tiedetään, kenen pelivuoro on, miten nappi muuttuu, ja kuhunkin nappiin on liitetty solu pelilaudalla.

Seuraavaksi toteutan voiton/häviön/tasapelin logiikan. Voitto tarkoittaa 3:ta samanlaista spritea joko diagonaalilla tai pystysuorassa. Käytän nappien matriisin indeksejä logiikan toteuttamiseen, sekä ehtolauseet.

2.2 Minimax-toteutus

Nyt toteutan automaattisen siirron logiikan, eli botti, joka käyttää minimax-algoritmia parhaan siirron löytämiseen. Kuvassa 4 on kuvioitu esimerkki pelin vaiheista, jotka riippuvat pelaajan valinnoista. Tila jossa voitto on todennäköisemmin on paras.



Kuva 4. Minimax-algoritmi kuvassa

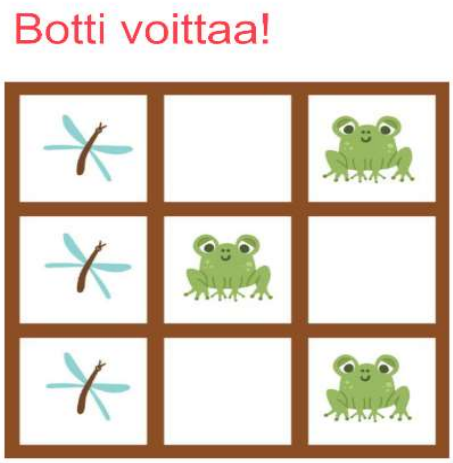
Löydetyn parhaan siirron tapauksessa, jos se on botilla, se palauttaa -1, jos pelaajalla, se on 1. Tasapelin tapauksessa se palauttaa 0. Tämän arvon perusteella botti tekee siirtonsa. Testauksessa se toimii melko nopeasti. Odottamani mukaisesti bottia ei voi voittaa nyt.

```
int MinimaxScore(bool isMaximizing)
{
    if (CheckForWin())
    {
        return isMaximizing ? -1 : 1; // Jos voitto, palautetaan -1 botin siirtoa varten, 1 pelaajan siirtoa varten
    }
    else if (CheckForDraw())
    {
        return 0; // Tasapeli
    }
```

Kuva 5. Pelin voiton arviointi

3 Yhteenveto

Peliin voi tutustua tässä. https://kaori-rei.itch.io/tictactoe
Peli on .zip-arkistossa ja latauksen jälkeen se voidaan käynnistää tietokoneella.
Pelikoodi löytyy minun GitHub repossa. https://github.com/arinasold/TicTac-Toe/blob/main/TicTacToeGame.cs



Kuva 6. Pelin kuvankaappaus

Oma oppimiskokemukseni tässä projektissa oli monipuolinen. Sain käsityksen Minimax-algoritmista ja sen soveltamisesta pelinkehityksessä. Lisäksi, pohdin pelisuunnittelun ja käyttäjäkokemuksen näkökulmia, kun integroin tekoälyn osaksi peliä. Työ antoi mahdolli-suuden reflektoida peliohjelmoinnin ja tekoälyn soveltamisen käytännön haasteita.

Jatkotutkimusmahdollisuuksia voisi olla tarkastella, miten erilaiset tekoälystrategiat suoriutuvat keskenään, kokeilla muita pelimuunnelmia Minimax-algoritmin kanssa, tai tutkia tapoja optimoida algoritmin suorituskykyä suuremmilla pelilautojen kooilla.

Lähdeluettelo

https://github.com/CoughBall/Aritifical_Intelligence_for_tictactoe

(GitHub public repo @CoughBall)

https://www.youtube.com/watch?v=hOcNZuQtQGE&ab_channel=OhMyCode

(“Oh My Code”, How To Make a Tic Tac Toe In Unity #03)

https://www.neverstopbuilding.com/blog/minimax

(Tic Tac Toe: Understanding the Minimax Algorithm)