Analýza zápočtové práce (Tic-Tac-Toe)

Úvod – Pravidla hry

Tic-Tac-Toe je hra pro dva hráče. Hraje se na čtvercové mřížce o rozměrech 3x3. Každý z hráčů má svůj symbol (obvykle to bývá křížek a kolečko), který střídavě vepisují do mřížky. V každém tahu označí hráč jedno volné políčko svým symbolem a pak nechá hrát druhého hráče. Hra končí vítězstvím v okamžiku, kdy jeden z hráčů vytvoří horizontální, vertikální nebo diagonální řadu se třemi svými symboly. Pokud už není možné vytvořit žádnou takovouto trojici, hra končí remízou.

Algoritmus umělé inteligence

Hra Tic-Tac-Toe má pro oba hráče neprohrávající strategii. Podle této strategie bude hrát i počítač, takže nikdy neprohraje.

V následujícím textu budu termínem *piškvorka* rozumět trojici polí umístěnou v řadě horizontálně, vertikálně nebo diagonálně. Termín *prázdná piškvorka* pak označuje piškvorku, která se skládá pouze z prázdných polí. *Moje piškvorka* představuje piškvorku, na které se nachází pouze moje značky a prázdná pole. Analogicky *soupeřova piškvorka*. Za *blokovanou piškvorku* budu považovat piškvorku, která obsahuje moje i soupeřovy značky.

Algoritmus umělé inteligence bude používat systém pravidel, podle kterých se rozhodne, kam by měl táhnout. Pravidla jsou následující:

1. Existuje moje piškvorka, ve které jsou již dvě pole označena, označím i třetí pole a tím zvítězím.
2. Existuje soupeřova piškvora, ve které jsou již dvě pole označena, označím třetí pole, abych zabránil soupeřovi vyhrát.
3. Existují dvě moje piškvorky, které se kříží na prázdném poli, označím si toto pole a tím získám dvě moje piškvorky se dvěma značkami, takže je soupeř nedokáže zablokovat.
4. Existují dvě soupeřovy piškvorky, které se kříží na prázdném poli:
    a. Pokud existuje moje piškvorka s jedním označeným políčkem, označím v ní druhé políčko a tím donutím soupeře, aby v příštím tahu blokoval.
    b. Táhnu na políčko, kde se soupeřovy piškvorky kříží, abych mu zabránil ve vytvoření dvojice piškvorek se dvěma značkami.
5. Zaberu si středové políčko.
6. Pokud má soupeř svoji značku v rohu, pak táhnu do protějšího rohu (je-li to možné).
7. Táhnu do volného rohu.
8. Táhnu na volné políčko uprostřed strany.

Tato pravidla se vyhodnocují ve výše uvedeném pořadí a algoritmus aplikuje první pravidlo, které lze použít.

Speciální část umělé inteligence tvoří zahájení (tj. první dva tahy). Vynecháme-li symetrie, existují tři možné způsoby, jak zahrát první tah: na střed, do rohu a na střed strany. Všechny tři tahy mohou vést k nucené remíze, avšak nejlepší volbou je zahrát do rohu, protože tento tah nabízí soupeřovi nejméně rozumných protitahů (musí odehrát na střed, jinak prohraje).

Druhý tah je snadný pokud soupeř zahrál do rohu, je potřeba zahrát střed a naopak (pokud soupeř zahrál střed je třeba zahrát do rohu). Pokud soupeř zahrál ke stěně, stačí nám zahrát střed, roh přilehlý k soupeřově tahu nebo střed protilehlé stěny. Jak je vidět, v druhém tahu už nám stačí dodržovat pravidla popsaná výše.

Další podrobnosti o hře a herních algoritmech naleznete například na stránce: http://en.wikipedia.org/wiki/Tic-tac-toe.

Implementace a uživatelské rozhraní

Hra bude napsána v jazyce Object Pascal ve vývojovém prostředí Delphi a používat bude pouze stadnardní knihovny VCL. Uživatelské prostředí bude tvořeno jednoduchými grafickými komponentami a hra bude ovládána myší.

Herní pole budou označována klasickými symboly křížek „X“ a kolečko „O“, přičemž za křížek hraje vždy první (začínající) hráč. V aplikaci půjde u každého hráče nastavit, zda za něho má hrát člověk, nebo umělá inteligence počítače. Lze tak vytvořit libovolné herní kombinace – včetně počítač vs počítač.

Rozdělení kódu

Kód bude rozdělen do dvou souborů. Herní algoritmy a datové struktury budou zapouzdřeny do samostatné unity, která bude mít čistě procedurální rozhraní, aby bylo možné ji použít i v klasickém (neobjektovém) Pascalu. Zde je krátký přehled nejdůležitějších procedur aplikačního rozhraní:

- **`procedure`** `resetGame;` - Inicializuje datové struktury na novou hru.
- **`function`** `onMove: char;` - Vrací symbol hráče, který je na tahu.
- **`function`** `getField(...): char;` - Vrací obsah hracího pole na daných souřadnicích.
- **`procedure`** `move(...);` - Zahraje tah na dané souřadnice.
- **`function`** `moveByAI;` - Nechá počítač zahrát tah a vrátí souřadnice, kam počítač zahrál.

Ve druhém souboru bude implementováno uživatelské rozhraní (formulář, grafické komponenty, reakce na události...).

Reprezentace dat

Herní plán bude v aplikaci uložen ve dvourozměrném poli 3x3.

Dále si bude aplikace pamatovat seznam všech (osmi) piškvorek, které se na herním plánu nachází. Každá piškvorka si pamatuje políčka, která ji vytváří, stejně tak každé políčko si pamatuje seznam piškvorek, které jím prochází.

Piškvorky jsou uloženy ve spojových seznamech (každá piškvorka leží právě v jednom), které zároveň určují stav piškvorky. Seznamy jsou následující:

- Prázdné piškvorky.
- Piškvorky prvního hráče.
- Piškvorky druhého hráče.
- Blokované piškvorky.

Tyto struktury mi usnadní implementaci algoritmu umělé inteligence, který je popsán výše.

Odhad časové a paměťové složitosti

Pravidla hry určují, že se Tic-Tac-Toe musí hrát na poli o rozměrech 3x3 a na žádném jiném. Z toho vyplývá, že časová i paměťová složitost bude konstantní.

Bohužel výše popsaný algoritmus nebude na větším poli fungovat, a tak nemá ani smysl odhadovat jeho časovou složitost pomocí parametrů. Paměťová složitost by pro větší pole byla $O(N \cdot M \cdot K)$, kde N a M jsou rozměry hracího pole a K je počet polí v řadě, která tvoří piškvorku.

Závěr

Pokud mi zbyde čas, rád bych ještě nějakým způsobem parametrizoval umělou inteligenci počítače, aby bylo např. možné hrát proti slabšímu počítači, který občas udělá chybu.