Doprovodný dokument k programu “Piškvorky”

1. Zadanie úlohy

Realizujte hru piškvorky. Hra se bude řídit standardními pravidly. Umožněte hru člověka proti počítači či hru dvou lidí proti sobě po síti. V případě hry proti počítači umožněte neomezené vracení se zpět v rozehrané partii.

2. Popis zvolených algoritmov

2. 1. Algoritmus samotnej hry

Pre samotnú hru som zvolil priamočiarý algoritmus. Po spustení sa užívateľ pomocou jednoduchého menu dostane k hre. V hre začína vždy užívateľ (v prípade sieťovej hry server). Po každom ťahu sa vyhodnocuje stav hry a uživateľ a jeho oponent sa postupne striedajú. Ak dôjde k výhre/prehre/remíze hra uvolní použité zdroje a v prípade sieťovej hry ukončí pripojenie. Následne sa hra ukončí.

2. 2. Algoritmus umelej inteligencie

Omnoho zaujímavejší je algoritmus umelej inteligencie. Ten upozorňujem nie je úplne dokonalý, ale zato pomerne šikovný, keďže dokázal neraz poraziť aj svojho tvorcu. Keby bol dokonalý nebolo by možné nad ním vyhrať, čo samozrejme nie je jeho účelom.

Algoritmus prebieha v troch fázach. Využíva dve pomocné polia, ktoré však nie sú zbytočné.

V prvej fáze sa počíta do prvého poľa hodnota určujúca ako veľmi je nutné ťahom obsadiť danú pozíciu, aby sa zabránilo užívateľovi vo výhre. Vyššia hodnota znamená vyššiu preferenciu. Hodnoty sú čísla od 0 do 100. K týmto hodnotám prídeme tým, že od daného políčka spočítame všetkými reálnymi smermi kontinuálne naväzujúce oponentské značky. Každý daný počet potom upravíme tak, že ho umocníme na 3. Tým sa nám znatelne oddelia dlhšie úseky a rozdiely medzi výslednými hodnotami. Po sčítaní týchto počtov sa teda daná hodnota uloží do daného prvého poľa na danú pozíciu.

V druhej fáze sa obdobne do druhého poľa rátajú značky umelej inteligencie. Tým nám vzniknú pole s obrannou a pole s útočnou preferenciou.

V tretej fáze sa hodnoty z oboch polí spočítajú a vyberie sa tá, ktorá má najváčššiu spoločnú preferenciu. Aby to nebolo také jednoduché a trochu reálne, musí sa počítač správať občas útočnejšie a občas obrannejšie. Preto sa hodnoty obrany a útoku násobia dvoma náhodnými desatinými číslami v rozsahu od 0 do 1. Tým sa počítač v rámci hry prikláňa náhodne k útočnej alebo obrannej stratégii.

Keďže políčka, ktoré už sú obsadené, majú preferenciu 0, nie je možné aby sa program pokusil obsadiť už obsadené pole.

3. Poznámky k implementácii

Program je implementovaný tak, aby nebol závislý na platforme. Je zkompilovateľný na ľubovoľnej platforme pomocou g++, avšak Makefile je robený výhradne pre školský Solaris, pretože využitie funkcií socketov si vyžiadalo pridanie parametrov pre linker -lnsl -lsocket. Pre využitie pod inou platformou je potrebné zameniť v súbore Makefile riadok:

LDFLAGS = -lnsl -lsocket

za:

LDFLAGS =

4. Záver

Program nebol až tak náročný, ako to, že rozsahovo som bol pred dokončením implementácie hrubo pod limitom. To mi sťažovalo prácu, ale nokoniec som to po konzultácii s prednášajúcim a po dokončení implementácie logiky umelej inteligencie dotiahol na spodnú hranicu.

Samo o sebe sa mi táto úloha písala pomerne dobre a rýchlo, asi kôli skúsenostiam zo semestra a progtestu. Problémom bol len Makefile, čo bola pre mňa novinka, ale po chvíľke surfovania po internete som aj tento posledný problém odstránil.

Všetko, čo nie je spomenuté v tomto dokumente, je obsiahnuté v komentároch v zdrojových súboroch.

5. Použité zdroje

Pri vývoji programu som čerpal z internetu a z publikácie Myslíme v jazyku C++.

1. Bruce Eckel (2000). *Myslíme v jazyku C++, knihovna programátora*. Grada Publishing, spol. s. r. o. Praha 7. ISBN 80-247-9009-2
2. Web predmetu Y36PSI: https://dsn.felk.cvut.cz/wiki/vyuka/y36psi/cviceni/prg_priklady
3. Ďalšie iné weby.