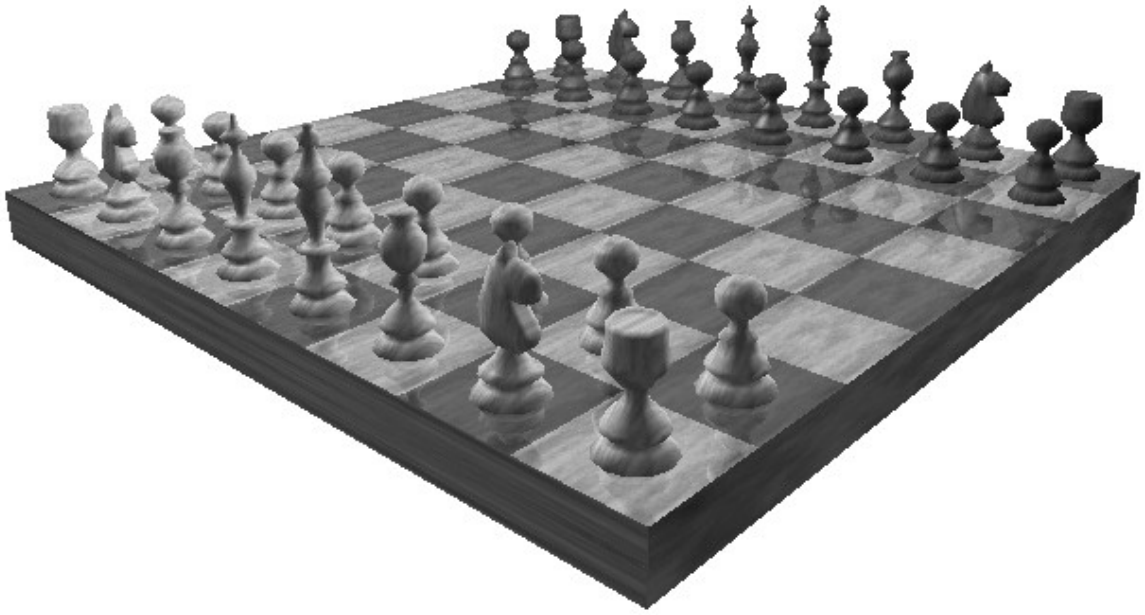
Ikt.sz:_____/_____

Bartók Béla Elméleti Liceum XII Matematika-Informatika

Informatika szakvizsgadolgozat

Chess 3D



Irányító tanár:
Mauzer Erika
Ilonczai Zsolt

Vizsgázó:
Purdea Andrei

Temesvár
2006

Tartalomjegyzék

1. Bevezető … 3

1.1 Miért fogtam neki … 3

1.2 Működéshez szükséges feltételek … 3

2. A Program Használata … 3

2.1 A Nézőpont Változtatása … 4

2.2 A sakkfigurák mozgatása … 4

3. Hogyan Készült … 4

3.1 A Sakktábla … 5

3.2 A Sakkfigurák … 5

3.3 A Mesterséges Intelligencia … 6

4. További Fejlesztési Lehetőségek … 6

5. Bibliográfia … 7

1. Bevezető

Szakvizsgadolgozatom témája egy jól kinéző, valósághű, három dimenziós sakkprogram készítése, amellyel lehet egy mesterséges intelligencia ellen játszani.

1.1 Miért fogtam neki

A programnak a nyári vakációban fogtam neki OpenGl grafika tanulása közben. Először csak a sakktábla készült el a sakkfigurákkal, később pedig a mesterséges intelligencia. Kezdetben a program elkészítésének a célja a tanult dolgok gyakorlása volt. Kezdetben a sakktábla elég jól sikerült, ezért úgy döntöttem, hogy fojtatom a fejlesztését.

1.2 Működéshez szükséges feltételek

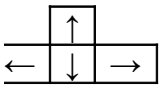
- IBM kompatibilis szamítógép, Apple számítógép, Palmtop, stb...
- Operációs rendszerek , amelyek képesek futtatni a programot: Linux, Windows, Windows CE, BeOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, és QNX. Ezek közül a programot lekompiláltam Linux-ra és Windows-ra. A többi operációs rendszeren nem teszteltem, de a cross platform fejlesztőeszköz szerint a tobbi platformon is gond nélkül lefut. (SDL Simple Directmedia Layer)
- Szükséges Merevlemezterület: ~600 KiloByte
- A gépen kell legyen működőképes OpenGl (min. 1 bit stencil buffer, 16 bit színmélység)
- Ajánlott egy 3D gyorsított videokártya használata, ellenkező esetben a program nagyon lassan fog működni. Fel kell legyen installalva a videokartya sajat drivere, lehetőleg a legújabb.
- A program egy AMD Sempron 2500+, 1,75GHz es gépen volt fejlesztve, melyben 256MB memória volt, és egy NVIDIA GeForce FX 5200 videokártya volt. A program ennél gyengébb rendszeren is kell hogy menjen, de biztosan nem állíthatom mivel nem próbáltam máshol.

2. A Program Használata

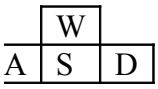
A programot a „./chess”(linux), vagy „chess.exe”(windows) lefuttatásával lehet elinditani. Linuxon a display környezet változó be kell legyen allítva ahhoz hogy a program hozzáférjen az X-hez. A Program a beindulás pillanatában egy animációval indul, amely körbejár a sakktábla körül , közben pedig közeledik és távolodik. Az animációt meg lehet állítani egy mouse click-el.

2.1 A Nézőpont Változtatása

Mivel a program három dimenziös, lehetőség van a nézőpont változtatására. A nézőpontot lehet a billentyűzettel változtatni, vagy a mouse-al. A billentyűzet használata több irányítást ad a felhasználó kezébe. A mouse használata viszont sokkal könnyebb, kényelmesebb.



A nyilak segítségével irányítható az X és Y tengelyeken való forgatás, vagyis az, hogy milyen dőlésszögben, és milyen oldalról lehet nézni a sakktáblát.
A Z és X gombok arra szolgálnak hogy be lehessen állítani hogy milyen távolságról legyen nézve a sakktábla.
A fenti nézőpont váltásokat el lehet végezni mouse-al is:
Jobb klikk – Drag and Drop-al lehet irányítani az X és Y tengelyeken a rotációt.
Középső gomb(vagy először jobb, és utánna bal gomb egyszerre, olyan mouse-ok esetén ahol nincsen csak 2 gomb) – Drag and Drop-al lehet irányítani a távolságot és az Y tengelyen a rotációt.
A Mouse Wheelt is lehet használni a távolság beállítására.
Eddig a Képernyő közepe teljesen megeggyezett a tábla közepével, az volt a “fókusz”-ban. A következő parancsokkal, ezt a pontot lehet mozgatni:



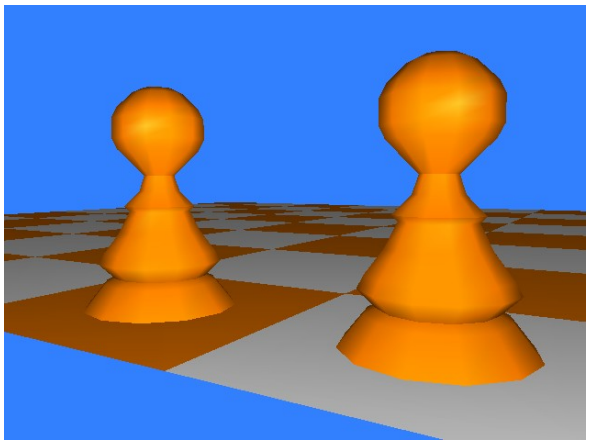
A W,A,S,D karakterek segítségével lehet előre,balra,hátra,jobbra járkálni a sakktáblán, és a Q,E gombokkal pedig lehet változtatni a “fókusz” magasságát.

2.2 A sakkfigurák mozgatása

A Sakkfigurák mozgatása a mouse-al történik, Bal klikk – Drag and Drop-al. A Klikkelt objektum lehet egy négyzet a sakktáblán vagy egy figura. A Bal gomb lenyomásának pillanatában pirossal bejelölődnek azok a négyzetek, ahova lehet mozgatni a kiválasztott figurát. A kiválasztott figura fel is emelkedik. A Kiválasztott figurát Drag and Drop-al a célra kell húzni, ami vagy egy kocka , vagy egy ellenséges figura. A Bal gomb eleresztésének pillanatában, amennyiben a lépés helyes, a figura az új helyére kerül, és a számítógép elkezdi kiszámítani a következő lépését. Amikor kiszámította végrehajtja a lépést, és fojtatódhat a játék.

3. Hogyan Készült

A program készítése három fő részből állt, a Saktábla, a Sakkfigurák, és a Mesterséges intelligencia elkészítése.
Jobbra látható egy a fejlesztés közben készült screenshot, itten még nem volt se textúrázás, se visszaverődés...

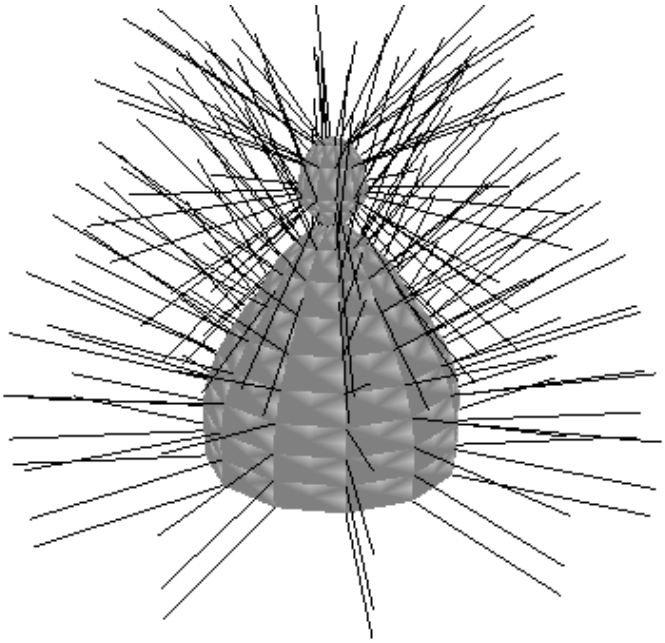


3.1 A Sakktábla

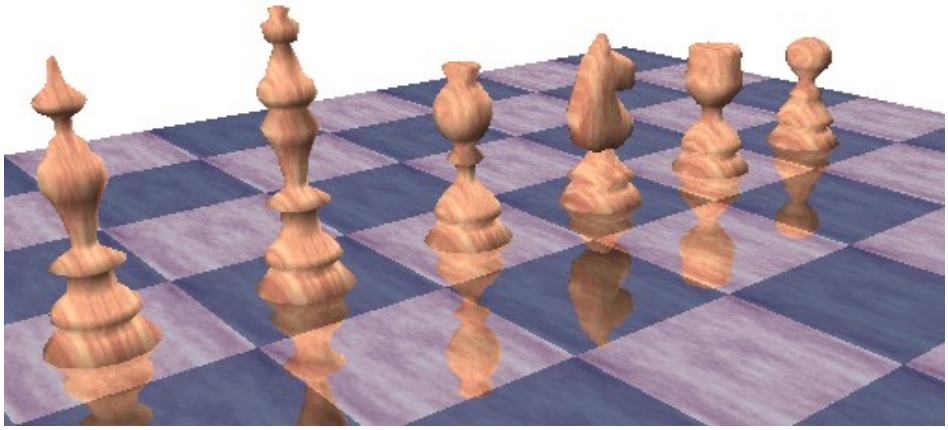
A sakktábla maga 8x8 + 5 = 69 QUAD-ból(négyoldalú felület) áll. A visszaverődés úgy volt megvalósítva, hogy először megrajzoltam a tábla feletti tárgyakat a tábla alá, és utánna rájuk rajzoltam a táblát félig átlátszóan, majd ujból megrajzoltam a tábla feletti tárgyakat a tábla felé. A fejjel lefelé rajzolást meg lehetett nagyon egyszerűen oldalni az y tengely megfordításával, amely egy függvényhívás: glScalef(1,-1,1); ezután még meg kell cserélni hogy az opengl milyen irányban rajzolt oldalakat tart előlapoknak, és a rajzolás helyesen történik.
Az, hogy a sakkfigurák tükörképei nem jelennek meg a sakktábla alján, az az OpenGl stencil bufferjának köszönhető. Ez az egyik dolog a sok közül amiért jobban megéri OpenGl-t használni, mivel DirectX-ben nincsen Stencil Buffer. Ebben annyi a lényeg, hogy csak oda rajzolok pixeleket az alsó figurákból, ahol meg fog jelenni a sakktábla. Ha lentről nézem a táblát akkor annak a felső síkja hátsó lap, és nem rajzolódik ki, így nem okoz vátozást a stencil bufferban, ezért nem rajzolódnak ki a figurák.
A sakktábla felett két fényforrás köröz, lassan, hogy ne zavarja a szemet.

3.2 A Sakkfigurák

Mint kezdõ modellezõnek a sakkfigurákat túl nehéz lett volna kézileg megcsinalni egy modellezőprogramban.
Ezért megrajzoltam kézzel a figurák körvonalait. Elhatároztam hogy hány pixelenként veszek egy felületet, és egy fill szerű algoritmussal bejártam a körvonalakat. Még meghatároztam két különböző színű pontot. Az egyik azt jelentette, hogy ott két felületre törtem egy hosszabbat, hogy élesebb, jobb minőségű legyen a figura, a másik pedig szakadási pontot jelentett. Ebben az esetben nem volt szabad átlagolni a normálvektorokat. Írtam egy dephi programot, amely beolvasta a körvonalat, és kiiratta egy fileba a forgástest háromdimenziós koordinátáit, és a normálvektorokat.
Jobbra látható egy kezdetleges paraszt a normálvektoraival. Ezt még a program tökéletesítésére használtam. Ezek után átalakítottam az adataimat Milkshape 3d formátumra, és ottan javítottam ki a figurák kisebb hibáit.
Ellentétben a többi figurával a ló nem szimmetrikus egy egyeneshez képest, így sajnos ezt kézileg kellett megcsinálnom.



A figurák most:



3.3 A Mesterséges Intelligencia

Szükség van ahhoz hogy lehessen játszani a számítógép ellen egy mesterséges intelligenciára. A jelenben a sakkprogramokra használt algoritmusok általában mind ugyanarra alapoznak. A táblán minden lehetséges figura-elhelyezésnek van egy értéke az egyik fél számára. A másik fél szempontjából ennek az értéke = minusz az egyik fél szempontjából vett értékre. Az algoritmus nagyjából egy mélységi keresésből áll, és arra az állításra alapszik hogy egy adott lépés értéke = minusz az ellenfél legjobb lépésének az értékével. Az alogirmus rekurzívan számolja az egymásutáni lépéseket. Ez természetesen egy végtelen algoritmus volna, ezért a rekurzív függvényhívásokat valahol le kell vágni. Ezért meg van határozva egy maximális mélység amibe a program bemegy. Amikor elérte az algoritmus a maximális mélységet akkor kell adjon az adott helyzetnek egy "statikus" értéket, amit a táblán levő figurák tipusából és számából számol ki.

Értéktáblázat:

Paraszt	100
Futó	325
Bástya	500
Királynő	900
Király	Végtelen
Ló	300

Az algoritmus végén a program végrehajtja a legjobb értékű lépést.
Az én programom 4-es mélységig keres, tehét 4 egymásutáni lépést számol ki.

4. További Fejlesztési Lehetőségek

A programot tovább lehet fejleszteni két irányban: Programot fel lehet szerelni hálózati képességekkel, és egyéb használatot segítő dolgokkal. Ezen kívül optimizálni lehet a mesterséges intelligenciát, azért hogy engedje meg hogy nagyobb mélységre keressen ugyanannyi idő alatt. Ezek az optimizációk ismert algoritmusok, példának fel lehet hozni Iteratív mélyítés, Alfa-Beta metszés, 64 bites bittáblák használata, stb....

5. Bibliográfia

[1] Simple Directmedia Layer http://www.libsdl.org/ , Dokumentacio: http://www.libsdl.org/docs.php

[2] The OpenGl Red Book

[3] NeHe OpenGl Tutorials http://nehe.gamedev.net/

[4] C++ konyvek es tutorialok a netrol: http://www.google.com/search?q=c%2B%2B

[5] Beowulf – Computer Chess Programming Theory - http://www.frayn.net/beowulf/theory.html