Miikka Haapasaari

KOLMIULOTTEINEN JÄTKÄNSHAKKIPELI

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2008

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Tekniikka ja liikenne	Koulutusohjelma Tietotekniikka
Tekijä(t) Miikka Haapasaari	
Työn nimi Kolmiulotteinen jätkänshakkipeli	
Vaihtoehtoiset ammattiopinnot Ohjelmistotekniikka	Ohjaaja(t) Olli Virmajoki
	Toimeksiantaja Kajaanin ammattikorkeakoulu
Aika 08.04.2008	Sivumäärä ja liitteet 31+1

Insinöörityö tehtiin Kajaanin ammattikorkeakoululle. Työn aiheena oli suunnitella ja ohjelmoida kolmiulotteinen jätkänshakkipeli. Ohjelma testattiin graafisella LCD-näytöllä. Näyttö oli kytketty mikrokontrolleri 8051:een. Ohjelma on yksi monista sovelluksista kyseiselle laitteistolle. Jätkänshakkipeliä voidaan käyttää esimerkkinä, kun esitellään kyseistä sulautettua laitteistoa.

Peli pelataan tietokonetta vastaan. Peli koostuu kolmesta 3 x 3 -peliruudukosta, jotka ovat näytöllä vierekkäin.Pelin säännöt ovat hyvin yksinkertaiset. Pelaajan tehtävänä on saada kolme omaa merkkiä peräkkäin. Rivin voi muodostaa jokaisella peliruudukolla tai jokaisen peliruudukon välillä. Pelaajan merkki on ”X” ja tietokoneen ”O”. Pelaaja aloittaa aina ensin. Pelaaja ja tietokone asettavat merkkinsä vuorotellen pelialustoille, kunnes voittaja on selvillä tai kaikki 27 ruudukkoa on täynnä.

Ohjelma kirjoitettiin C-ohjelmointikielellä, ja testialustana käytettiin valmista sulautettua järjestelmää. Kyseinen alusta sisältää mikrokontrolleri 8051:n, LCD-näytön ja matriisinäppäimistön. Ohjelmointiympäristönä toimi IAR ANSI C -kääntäjä ja testausympäristönä EMUL 51 -emulaattori.

Jätkänshakkiohjelma koostuu kahdesta ohjelmistomoduulista. Pelin pääohjelma sisältää pelin toiminnallisuuden ja tekoälyn. LCD-ohjelma sisältää alustusrutiinit ja graafiset toiminnallisuudet. Ohjelmistomoduuleihin kuuluvat myös tarvittavat otsikkotiedostot, jotka sisältävät aliohjelmien esittelyt ja ohjelmistomoduuleihin liittyviä määrittelyjä.

Sovellus testattiin eri ohjelmistovaiheissa. Testauksessa testattiin testilaitteiston toiminnan kannalta tärkeä ohjelma, pääohjelma ja tekoäly. Lopputulokseksi saatiin laitteistolle testauksien jälkeen toimiva ohjelma.

Kieli	Suomi
Asiasanat	Kolmiulotteinen jätkänshakki, 8051, C-kieli
Säilytyspaikka	☐ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☐ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School Engineering	Degree Programme Information Technology
Author(s) Miikka Haapasaari	
Title 3D Tic-Tac-Toe Game	
Optional Professional Studies Software Engineering	Instructor(s) Olli Virmajoki
	Commissioned by Kajaani University of Applied Sciences
Date 8 April 2008	Total Number of Pages and Appendices 31+1

This bachelor's thesis was commissioned by the Kajaani University of Applied Sciences. The purpose of the thesis was to design and program the code of a 3D Tic-Tac-Toe game. The program was tested on a graphical LCD display. The display was connected to a 8051 microcontroller. The software is only one of the many applications for this device. The Tic-Tac-Toe game can be used to demonstrate the functionality of the embedded hardware.

The game is played against a computer player. The game consists of three 3 x 3 boards which are on the display side by side. The rules are very simple. The player goal is to be the first to get three crosses in a row. The row can be on any of the three boards or between three boards. The player is known as X and the computer is O. The player always goes first. The player and the computer alternate placing symbols on the game boards until either one has three symbols in a row or all twenty-seven squares are filled.

The program was made by using the C language. The existing embedded hardware was used as the testing hardware. The hardware consists of the 8051 microcontroller, an LCD display and a matrix keyboard. The IAR ANSI C complier was used as the software development environment and EMUL 51 was used as the testing environment.

The Tic-Tac-Toe program consists of two program modules. The main program contains the functionality of the game and artificial intelligence. The LCD display control program consists of the graphical drivers and functionality. The modules include header files and the programmer definitions.

The program was tested during the different phases. The testing contained the program required by the hardware and the main program of the game with artificial intelligence. The result after testing was working software.

Language of Thesis	Finnish
Keywords	3D Tic-Tac-Toe, 8051, C language
Deposited at	☐ Kaktus Database at Kajaani University of Applied Sciences ☐ Library of Kajaani University of Applied Sciences

ALKUSANAT

Tämä insinöörityö tehtiin Kajaanin ammattikorkeakoululle. Haluan kiittää suuresti valvojaani Olli Virmajokea ja työn tilaajaa Ismo Talusta Kajaanin ammattikorkeakoulun puolelta. Haluan myös kiittää avopuolisoani Minna Niemelää tuestaan. Hänen tuellaan on ollut suuri vaikutus työn valmistumiseen.

Pyhäjärvellä 8.4.2008

SISÄLLYS

SYMBOLILUETTELO JA TERMIEN SELITYS

B-kieli	AT & T: Bell Labsin kehittämä ohjelmointikieli. Nykyään se on kuollut lähes sukupuuttoon, koska C-kieli on korvannut sen.
CPU	Tarkoittaa (eng. Central Processing Unit) tietokoneen suoritinta eli prosessoria. Sen tarkoituksena on toimia tietokoneen sydämenä. Sen tehtävänä on suorittaa tietokoneohjelman sisältämät konekäskyt.
Kirjasto	Tietotekniikassa kirjastot ovat ohjelmia, joita käytetään tietokoneohjelmien kehittämisessä sekä ohjelmien suorittamisen aikana.
RAM	RAM-muisti (eng. Random Access Memory). Keskusmuisti tai käyttömuisti on tietokoneohjelmien työmuisti, johon latautuvat käyttöjärjestelmän ohjelmat, suoritettavat sovellukset sekä näiden tarvitsemat tiedot. Keskusmuistin sisältö tyhjenee aina virrankatkaisun yhteydessä.
ROM	ROM-muisti (eng. Read-Only Memory), lukumuisti on tietokoneen pysyvä muisti, johon ei voi tehdä muutoksia normaalin käytön aikana. Tiedot säilyvät, vaikka tietokone sammutetaan.
Termi	Selvitys
UML	UML-mallinnus (eng. Unified Modeling Language) on OMG vuonna 1997 standardoima graafinen mallinnuskieli, joka sisältää 13 erilaista kaaviota.
UNIX-käyttöjärjestelmä	Laitteistoriippumaton käyttöjärjestelmä, jonka kehityksen aloittivat AT&T:n Bell Labsin työntekijät vuonna 1969.

1 JOHDANTO

Työ tehtiin Kajaanin ammattikorkeakoululle. Kajaanin ammattikorkeakoulu tarjoaa 2000 oppilaalle opiskelupaikan. Kajaanin ammattikorkeakoululla voidaan suorittaa sairaanhoitajan, terveydenhoitajan, insinöörin, restonomin, tradenomin ja liikunnanohjaajan tutkinnon. Koulun vahvuutena pidetään yhteistä kampusaluetta, opinto-ohjausta sekä tiivistä yhteistyötä kajaanilaisten yritysten kanssa.

Työni liittyy läheisesti Kari Niskasen tekemään insinöörityöhön [1.]. Testilaitteiston osalta työmme ovat samanlaisia ja perusideakin on aika sama. Niskanen keskittyi työssään lähinnä ”itseoppivan” tekoälyn kehittämiseen, kun taas minun työssäni tekoälyllä ei ole niin suurta merkitystä. Minä puolestaan keskityin luomaan kolmiulotteisen pelialustan eli pelialustoja on kolme. Niskasen peliautomaatti toimii yhdellä pelialustalla.

Insinöörityön tarkoituksena on ohjelmoida perinteisestä jätkänshakkipelistä kolmiulotteinen versio. Pelissä on kolme 3x3 -peliruudukkoa, joista mahdollinen voitto tulee. Voiton saavuttaa, jos saa kolme merkkiä peräkkäin vaaka-, pysty- tai syvyyssuunnassa. Voiton syvyyssuunnassa saavuttaa esimerkiksi asettamalla oman merkin jokaisen peliruudukon vasempaan yläreunaan. Ohjelmalle luodaan tekoäly, joka on aika yksinkertainen. Kokeneen pelaajan on helppo voittaa tietokone.

Ohjelmointi suoritetaan C-kielellä. Vaihtoehtona oli myös asembly-kieli, mutta päädyin C-kieleen, koska sitä on mielestäni helpompi käyttää ja ymmärtää. Valmis ohjelma tulee siirtää datakaapelilla valmiiseen sulautettuun testilaitteistoon.

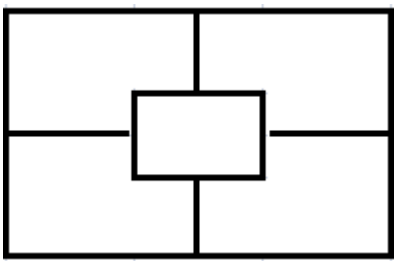
Kolmiulotteisuus on toteutettu graafisella näytöllä siten, että luodaan kolme 3x3-peliruudukkoa vierekkäin. Tähän ratkaisuun päädyin, koska C-kielellä on erittäin vaikea toteuttaa 3D-grafiikkaa. Ensimmäiseksi luodaan kuitenkin itse peli, ja vasta sen jälkeen panostetaan graafiseen puoleen. Laitteen tekniset osat ja toimivuus pitää myös selvittää siltä osin, kuinka ne vaikuttavat ohjelman siirtämiseen ja toimivuuteen näytöllä.

Aluksi insinöörityössä kerrotaan tavallisen jätkänshakkipelin historiasta ja säännöistä. Kolmiulotteisen version rakenne ja säännöt on kerrottu seuraavassa luvussa. Tämän jälkeen on kerrottu testilaitteistosta. Viimeiseksi kerrotaan ohjelman toiminnallisuudesta ja käydään läpi ohjelman testaus.

2 JÄTKÄNSHAKKIPELIN SÄÄNNÖT JA HISTORIA

Jätkänshakki eli ristinolla on ollut lasten suosima peli jo useiden sukupolvien ajan. On erittäin vaikeaa tarkentaa, milloin ristinolla on keksitty. Rivipelejä, kuten ristinolla, on pelattu tuhansia vuosia. Rivipeleissä ideana on asettaa pelinappuloita pelilaudalle siten, että muutama niistä muodostaa rivin. Egyptiläisen Al-Qurnan temppelin kattolaatoista on löydetty rivipeleissä käytettyjä pelialustoja. Nämä alustat on tiettävästi kaiverrettu kiveen 3 500 vuotta sitten. Kiinassakin pelattiin yksinkertaista rivipeliä nimeltä ”yi” jo 500 eKr. [2, s. 103.]

Ristinollaa pidetään myllypelin muunnelmana. Myllypeli (ks. kuva 1) on kahden hengen peli, jossa kummallakin on yhdeksän pelinappulaa. Pelaajat asettavat vuoron perään nappuloita yksitellen laudalle. Yrityksenä on saada kolme nappulaa peräkkäin samalle suoralle eli saada ”mylly”. Kiinassa tämä peli tunnettiin nimellä ”yi” ja sitä pelattiin 500 vuotta ennen ajanlaskumme alkua. [2, s. 103–104.]



Kuva 1. Tavallinen myllypelialusta.

Ristinollaa pelataan ruudukolla. Pelin aloittaja piirtää ristin johonkin yhdeksästä ruudusta, ja toinen pelaaja puolestaan piirtää ympyrän johonkin toiseen ruutuun. Pelaajat piirtävät merkkejään vuorotellen. Merkin paikkaa ei voi vaihtaa, kun siirto on tehty. Pelin voittaa se, joka ensimmäisenä saa kolme omaa merkkiään samalle riville, vaaka- tai pystysuoraan tai viistoon. Toinen pelaaja voi voittaa vain, jos aloittaja tekee virheen [2, s. 103.]. Ristinolla on hyvin suosittu lasten keskuudessa, vaikka se on yksinkertainen peli.

3 KOLMIULOTTEISEN VERSION RAKENNE JA SÄÄNNÖT

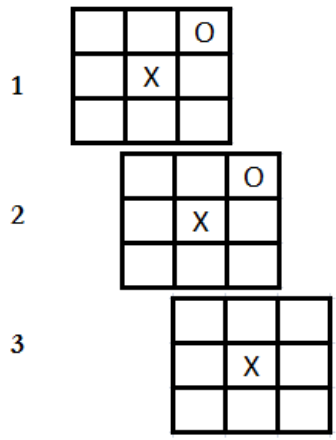
Pelin ideana on pelata tietokonetta vastaan, eli tässä tapauksessa luodaan ohjelmalle keinotekoinen tekoäly. Pelin säännöt ovat yksinkertaiset:

- Peli koostuu kolmesta tasosta (ks. kuva 2). Jokainen taso koostuu kolmesta pystysuorasta ruudusta ja kolmesta vaakasuorasta ruudusta. Yhteensä ruutuja tulee olemaan 27 graafisella näytöllä.

- Kaksi pelaajaa vuorotellen merkkaavat ruutuun joko X:n tai O:n. Voittajan on saatava kolme omaa merkkiä peräkkäin pystysuoraan, vaakasuoraan, vinoon tai kolmiulotteisessa versiossa syvyyssuunnassa. Tässä jätkänshakkipelissä pelaaja aloittaa aina ensin. Merkin paikkaa ei voi muuttaa, kun siirto on tehty.

- Pelaaja siis voittaa, jos X-merkit muodostavat kolmen suoran. Suoran ei tarvitse olla pelkästään yhdellä tasolla. Sen voi muodostaa useammalla tasolla, koska kyseessä on kolmiulotteinen peli.

Peli on säännöiltään aivan samanlainen kuin tavallinen jätkänshakki. Verrattuna tavalliseen versioon kolmiulotteisessa on helpompi muodostaa kolmen suora, koska peli jakaantuu kolmelle eri tasolle. Kolmiulotteinen versio ei ole niin tylsä kuin tavallinen jätkänshakki, koska siinä tasapelin todennäköisyys on pienempi. Kolmiulotteinen versio vaatii pelaajalta enemmän strategista kykyä ja on siten kiinnostavampi.



Kuva 2. Kolme tasoa muodostaa kolmiulotteisen jätkänshakkipelin.

4 KÄYTETTÄVÄ OHJELMOINTIKIELI, LAITTEISTO JA YMPÄRISTÖ

Jätkänshakkipeli on kirjoitettu C-ohjelmointikielellä. Ohjelma on toteutettu valmiiseen testilaitteistoon. Testilaitteisto sisältää graafisen LCD-näytön (Hitachi LM214XB), mikrokontrolleri 8051:n ja 4x4-matriisinäppäimistön. Kuvassa 3 on esillä työssä käytetty testilaite, johon ohjelma ladattiin.

Kuva 3. Työssä käytetty testilaite.

4.1 C-kieli

Dennis Ritchie kehitti C-ohjelmointikielen USA:ssa AT & T:n Bellin laboratorioissa vuonna 1972. C-kieltä kehitettäessä käytettiin pohjana B-kieltä. [3, s. 1.]

C-kieli on alun perin syntynyt UNIX-käyttöjärjestelmää suunniteltaessa, ja siksi siinä yhdistyvät koneläheisen ja korkean tason ohjelmointikielen ominaisuudet. C-kieli on joustava, joten se ei rajoita käyttäjän työskentelyä ja on siksi sopiva tämän työn ohjelmointikieleksi.

Huonoa C-kielessä on se, että se ei ohjaa tai valvo ohjelmointia. Ohjelmoijan täytyy olla kohtuullisen hyvin perillä C-kielen ohjelmointityylistä, standardeista ja ohjelmointitekniikoista käyttääkseen sitä työkaluna ohjelmointiprojekteissa. [3, s. 1.]

C-kieltä ei ole suunniteltu sovellusohjelmointikieleksi. Sovellustoiminnot on toteutettu erilaisilla työkaluilla. Työkaluajattelu ja uudelleenkäytettävyys ovat keskeisiä asioita C-kielessä. Työkalut toteutetaan käytännössä aliohjelmilla. Näitä voidaan tehdä itse tai hankkia valmiiksi standardoituja aliohjelmakirjastoja. [3, s. 2.]

C-kieli rakentuu itse kielestä, esikääntäjästä ja standardikirjastosta. Esikääntäjä muokkaa lähdetekstiä ennen varsinaista C-käännöstä. Esikääntäjä on yleensä integroitu varsinaiseen kääntäjään niin, että esikäännös tapahtuu aina automaattisesti ilman eri toimenpiteitä. Standardikirjasto sisältää standardoidun joukon valmiita aliohjelmia. [3, s. 2.]

C-kieli on oiva työkalu ohjelmoijalle, joka on perehtynyt C-kieleen ja yleisesti ohjelmointiin. Aloittelijalle se tuottaa suuria ongelmia, jos hän yrittää tehdä isompia ohjelmointiprojekteja ilman aikaisempaa kokemusta C-ohjelmoinnista.

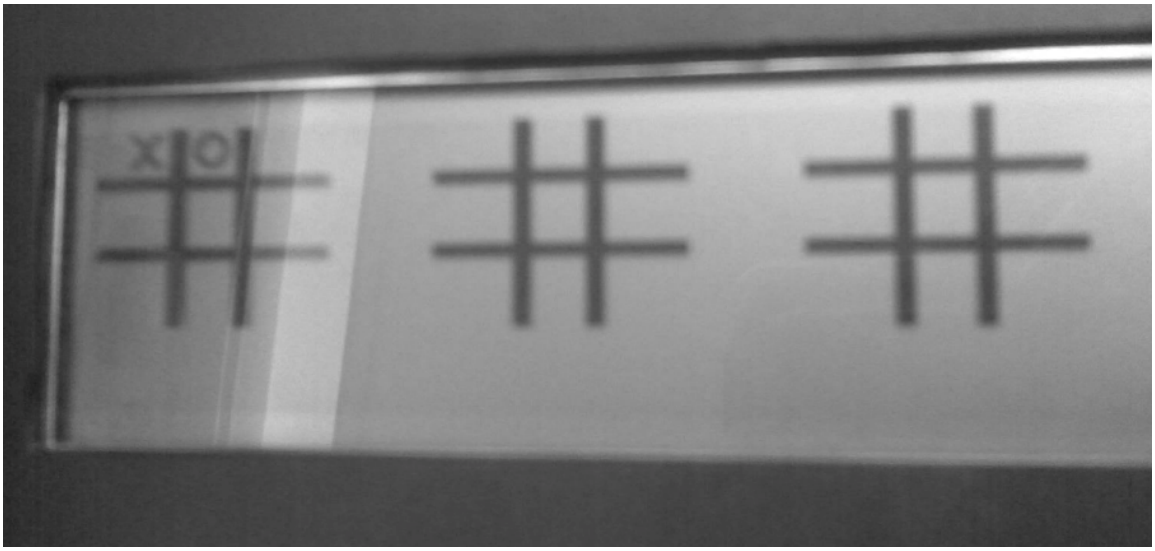
4.2 Mikrokontrolleri 8051

Mikrokontrolleri 8051 kuuluu Micro Controller System-51 -piiriperheeseen (MCS-51), jonka on suunnitellut Intel. Tosin nykyään Intel ei valmista kyseiseen piiriperheeseen kuuluvia piirejä. Tähän piiriperheeseen kuuluu useita piirejä. Piirit eroavat toisistaan kolmella eri tavalla: piirille integroidun muistin määrän, tyypin ja piirille integroitujen I/O-ominaisuuksien puolesta. Yhteistä kaikille piireille on se, että ne omaavat saman käskykannan. 51-piiriperheen 8051-piirin pääominaisuuksia ovat:

- 8-bittinen keskusyksikkö
- 64 kilotavun muistiavaruus ohjelmamuistille
- 64 kilotavun muistiavaruus ulkoiselle datamuistille
- 4 kilotavua sisäistä ohjelmamuistia eli ROM-muistia
- 128 tavua sisäistä datamuistia eli RAM-muistia
- kaksi 16-bittistä ajastinta/laskuria
- sisäinen kello-oskillaattori
- vektoroitu keskeytysrakenne, jossa kaksi prioriteettitasoa
- laaja käskykanta: kuusi osoitusmuotoa, yksittäisen bittien osoitusmahdollisuus
- täysin kaksisuuntainen sarjaportti
- 32 kaksisuuntaista ja yksittäin osoitettavaa I/O-linjaa [4, s. 17.]

4.3 Graafinen LCD-näyttö

Graafisena näyttönä toimii Hitachi LM213XB vihreä LCD-näyttö. Näyttö toimii tekstitilassa sekä graafisessa tilassa. Näytön ohjaus tapahtuu integroidun HD61830-ohjainpiirin avulla. Piirin kapasiteetti grafiikan osalta on 2^{16} bittiä eli 512k pistettä ja tekstitilan kapasiteetti on 4096 merkkiä [5.]. Ohjainpiiristä löytyy omaa näyttömuistia (VRAM) ja sisäänrakennettu merkkigeneraattori (CGROM). Kuvassa 4 näkyy laitteiston graafinen LCD-näyttö.



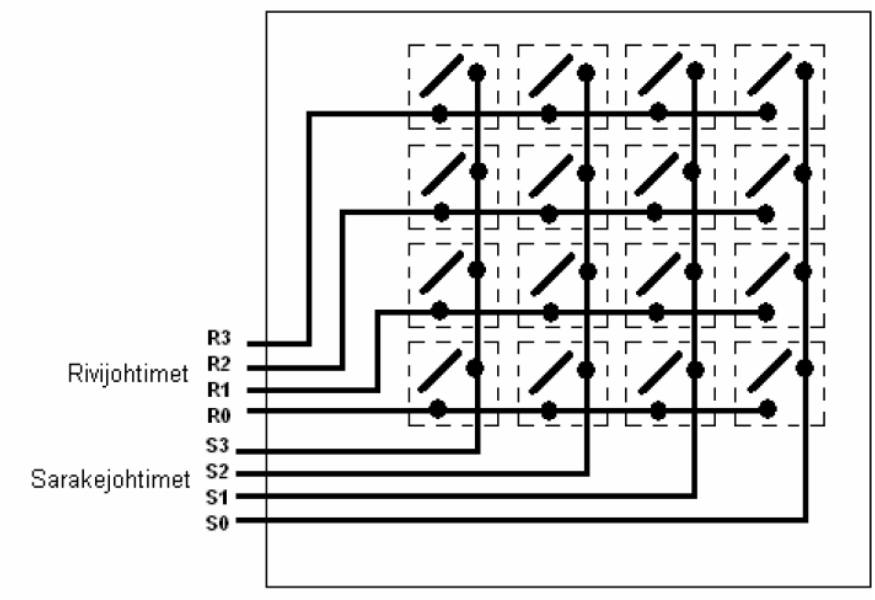
Kuva 4. Hitachi LM213XB:n graafinen LCD-näyttö.

4.4 Matriisinäppäimistö

Laitteistossa on käytetty 4x4-matriisinäppäimistöä, joka näkyy kuvassa 5. Näppäimistö sisältää seuraavat näppäimet: 1, 2, 3, 4, 5, 6, 7, 8, 9, F, E, D ja C.

Kuva 5. Matriisinäppäimistö. [1.]

Yksittäinen näppäin näkyy CPU:lle omana bittinä, joka voidaan käydä lukemassa. Matriisinäppäimistöllä jokaiselta painonappulalta ei saada omaa erillistä signaalia. Näppäimistöllä näppäimen painaminen tapahtuu siten, että yksi rivi- ja sarakejohdin menevät keskenään kontaktiin, kuten kuvassa 6 näkyy. [4, s. 86.]



Kuvassa 6. 4x4-matriisinäppäimistön sisäinen kytkentä. [4, s. 87.]

Näppäimistöllä ei ole valmiina käyttöjännitettä, joiden perusteella voitaisiin päätellä loogiset ”0”- ja ”1”-tilat. Matriisinäppäimistöltä ei saada suoraan yhdellä lukemisella tietää, mitä näppäintä on painettu. Painettu näppäin saadaan selville, kun ajetaan näppäimistölle sarja kirjoitus- ja lukujaksoja. Näillä jaksoilla käydään läpi koko näppäimistö rivi tai sarake kerrallaan. [4, s. 87.]

Näppäimistö pitää skannata, jotta painettu näppäin saadaan selville. Skannaus tapahtuu siten, että rivijohtimiin (ks. kuva 6) kirjoitetaan yhteen kerralla ”1”-tila, joka siis vastaa käyttöjännitettä. Sarakejohtimia (ks. kuva 6) lukemalla saadaan selville, että minkä rivin ja sarakkeen risteyskohdassa painettu näppäin oikein on. [4, s. 87.]

Painettu näppäin saadaan selville, kun kirjoitetaan rivijohtimeen ensin sana 0001, sitten 0010, seuraavaksi 0100 ja lopuksi 1000. Jokaisen kirjoituksen jälkeen luetaan sarakejohtimelta näppäimen tila. Neljän kirjoitus- ja lukujakson jälkeen ohjelmistolla pystytään päättelemään, mikä näppäin on painettuna. [4, s. 87.]

Kehittyneissä matriisinäppäimistökytkennöissä CPU:n ei tarvitse tehdä turhia näppäimistöskannauksia, jos mitään näppäintä ei ole painettu. Näppäimistön antama hälytyslinja kytketään CPU:n keskeytystuloon ja skannaus tehdään keskeytysaliohjelmassa. [4, s. 87.]

4.5 Ohjelmointiympäristö

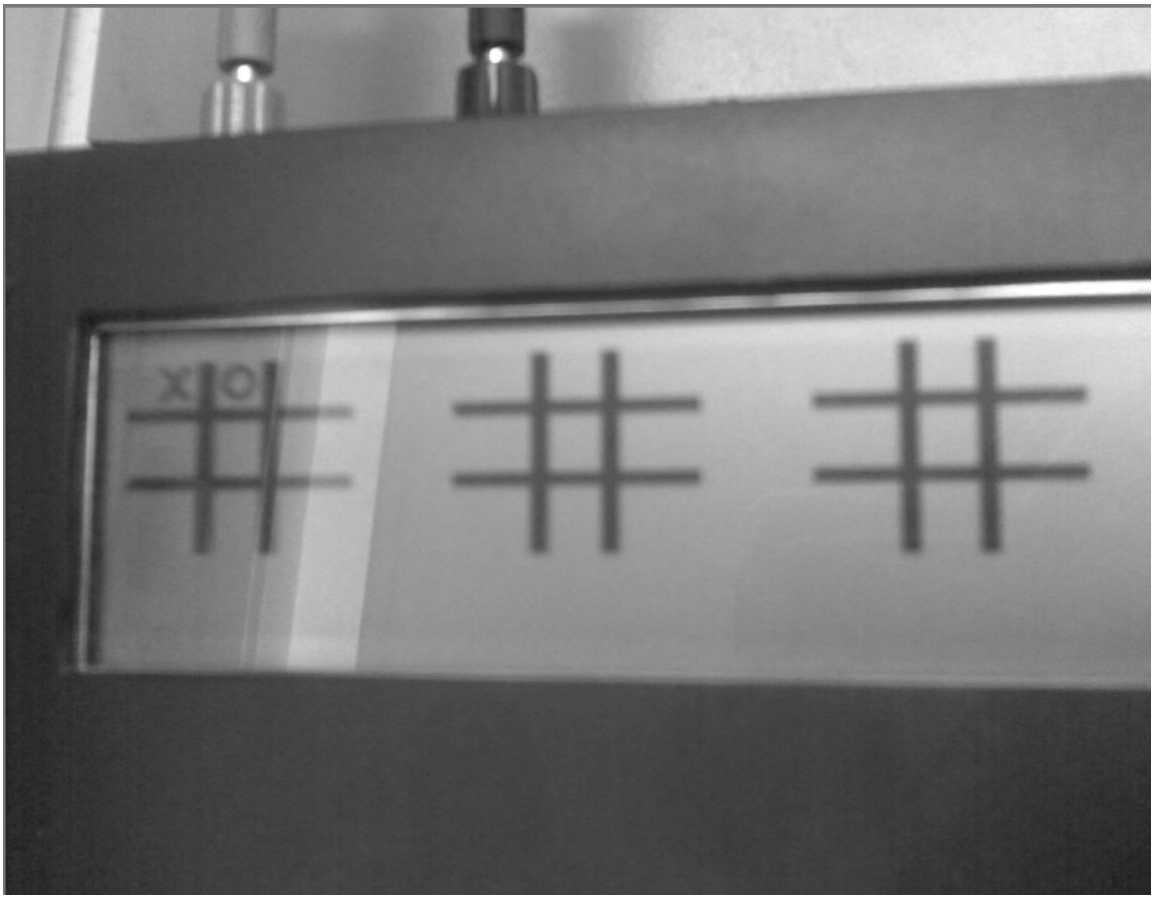
Ohjelma kirjoitettiin C-kielellä, ja ohjelmointiympäristönä ohjelmoinnissa käytettiin PC-tietokonetta ja IAR ANSI C-kääntäjää. Ohjelman testaukseen testilaitteessa käytettiin EMUL51-emulaattoria. Ohjelman suunnittelu- ja testaustyö tehtiin Kajaanin ammattikorkeakoulun tietotekniikan laboratoriossa. Koululta löytyivät kaikki tarvittavat ohjelmistot sekä laitteet sovelluksen suunnittelua ja testausta varten. Ohjelman koodin kirjoittaminen tehtiin osaksi myös omalla kotikannettavalla, jossa käytettiin ohjelmointiympäristönä Microsoft Visual Studio 2005:sta ja IAR Embedded Workbench Kickstart Edition 4.0 -versiota.

IAR ANSI C-kääntäjä (IAR Embedded Workbench) muuttaa korkean tason kielen lähdekoodin mikrokontrollerille sopivaksi binäärikoodiksi, jotta se on mahdollista ladata PC:ltä laitteeseen. EMUL51-emulaattori puolestaan mahdollistaa täydellisen mikrokontrollerin hallinnan. Tällöin laitteiston ohjaimen kaikki portit ja muistit ovat käytettävissä.

5 KOLMIULOTTEISEN JÄTKÄNSHAKKIPELIN TOTEUTUSRATKAISUT

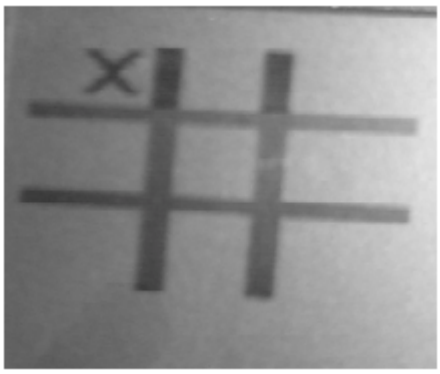
5.1 Yleiskuvaus ohjelmasta

Ohjelman alussa ajetaan LCD-näytölle tarvittavat alustusrutiinit eli alustetaan graafinen näyttö jätkänshakkiohjelmaa varten. Alustusrutiinien jälkeen näytölle piirtyy vierekkäin kolme 3x3-peliruudukkoa (ks. kuva 7). Alustuksen ja ruudukkojen piirtämisen jälkeen käynnistyy vasta virallinen peli. Pelaaja aloittaa aina ensimmäisenä pelin. Tietokone yrittää ensisijaisesti estää pelaajan kahden suorat. Vaikeustaso koko pelin aikana tulee olemaan helppo. Aloittelijalle saattaa tulla vaikeuksia tietokoneen kanssa. Kokeneelle pelaajalle ei tuota ongelmia voittaa tietokone.



Kuva 7. Kolme 3x3-peliruudukkoa LCD-näytöllä.

Ohjelman käynnistyessä ruudulle ilmestyy pelin nimi ja tekijä. Pelaajaa käsketään aloittamaan peli painamalla F-näppäintä. Seuraavaksi pelaaja valitsee matriisinäppäimistön A-, B- tai C-näppäimellä, mihin ruutuun hän haluaa tehdä siirron. Seuraavaksi pelaaja valitsee numeronäppäimillä 1−9, kohdan mihin hän haluaa tehdä siirron. Eli jos pelaaja painaa esimerkiksi A:ta ja sen jälkeen ykköstä, niin pelaajan merkki ”X” piirretään ensimmäisen ruudukon vasempaan yläreunaan (ks. kuva 8). Pelaajalla on aina merkki ”X” ja tietokoneella (CPU) on merkki ”O”. Peliruudukot tarkastetaan, kun molemmat ovat tehneet siirtonsa, onko voittaja selvillä eli onko toinen saanut kolmen merkin suoran. Peli päättyy, kun voittaja on selvillä tai on päädytty tasapeliin. Tasapeliin päädytään, jos voittaja ei ole selvillä 27 siirron jälkeen. Siirtojen moninkertaisuus verrattuna tavalliseen jätkänshakkipeliin johtuu siitä, että ruudukkoja on pelissä kolme.



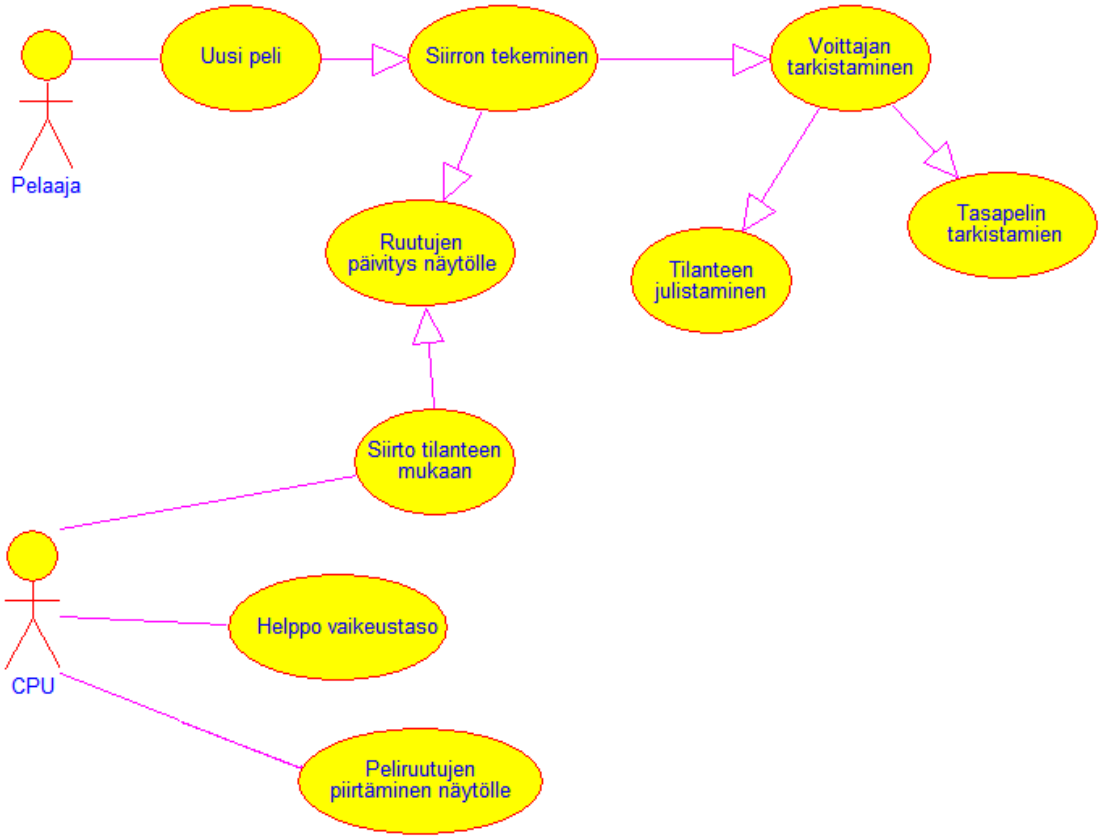
Kuva 8. X-merkki ruudukon vasemmassa yläreunassa.

Tietokone ilmoittaa erän päätyttyä voittajan tai julistaa tasapelin. Peli alkaa aina alusta. Tietokoneella on siis vain yksi tekoäly. Vaikeustasoa kolmiulotteisessa jätkänshakissa voidaan pitää helpohkona. Ohjelman toiminnallinen vuokaavio on esitetty liitteessä 1.

Ohjelma on jaettu kahteen osioon: TIC.c (pääohjelma) sisältää pelin toiminnallisuuden ja LCD.c hoitaa graafisen näytön ohjauksen ja sekä alustusrutiinit. Lisäksi ohjelmistosta löytyy kaksi otsikkotiedostoa TIC.h ja LCD.h. Edellä mainitut tiedostot sisältävät ohjelmistoihin liittyviä määrittelyjä.

5.2 Vaatimusten määrittely

Kuvassa 9 on kuvattu kolmiulotteisen jätkänshakkipelin toiminnallisuus käyttötapauskaavion avulla. Kaaviolla on tarkoitus kuvata, miten ohjelmassa edetään pelaajan valintojen ja tiettyjen ohjelmatoimintojen jälkeen. Kaavio on tehty Prosa/om UML modeller -ohjelmalla, joka on tarkoitettu UML-kaavioiden tekemiseen.



Kuva 9. Kolmiulotteisen jätkänshakkipelin käyttötapauskaavio.

Alla olevissa taulukoissa 1−9 on esitetty jätkänshakkipelin käyttötapauslomakkeet.

Taulukko 1. Uusi peli.

Käyttötapaus:	Uusi peli
Aktorit:	Pelaaja
Ehdot:	Ohjelma on käynnistetty. Kaikki tarvittavat alustukset on suoritettu. Ohjelma kysyy pelaajalta pelin aloittamista.
Kuvaus:	Ohjelma käskee pelaajaa aloittamaan pelin. Pelaaja aloittaa pelin painamalla näppäintä F.
Poikkeukset:	Ohjelma ei käynnisty. Näytön alustus ei onnistu. Edellinen peli on käynnissä.
Lopputulos:	Ohjelma käskee pelaajaa aloittamaan pelin.

Taulukko 2. Peliruutujen piirtäminen näytölle.

Käyttötapaus:	Peliruutujen piirtäminen näytölle
Aktorit:	CPU
Ehdot:	Ohjelma on käynnistetty. Kaikki tarvittavat alustukset on suoritettu.
Kuvaus:	Ohjelma piirtää kolme vierekkäistä 3x3 -peliruutua näytölle.
Poikkeukset:	Näyttö ei alustu tai peli ei käynnisty.
Lopputulos:	Näytölle tulostuu kolme 3x3 -peliruutua.

Taulukko 3. Siirron tekeminen.

Käyttötapaus:	Siirron tekeminen
Aktorit:	Pelaaja
Ehdot:	Tekee siirron, jos siirtoja on jäljellä ja on pelaajan oma vuoro kyseessä.
Kuvaus:	Pelaaja tekee siirron haluamaansa kohtaan, joka päivittyy haluttuun peliruudukkoon.
Poikkeukset:	Siirtoja ei ole, eli peli on loppunut. Vuoro on CPU:lla.
Lopputulos:	Pelaaja on tehnyt siirron, ja se on päivittynyt haluttuun peliruudukkoon. Vuoro siirtyy CPU:lle.

Taulukko 4. Ruutujen päivitys näytölle.

Käyttötapaus:	Ruutujen päivitys näytölle
Aktorit:	CPU
Ehdot:	Peli on alkanut ja siirto on tehty.
Kuvaus:	Piirretään merkki "X" tai "O" ruudukkoihin, riippuen siirron tekijästä.
Poikkeukset:	Näytön alustus ei onnistu. Siirtoja ei ole jäljellä. Siirtoa ei ole tehty.
Lopputulos:	Merkki "X" tai "O" päivittyy näytölle valittuun kohtaan.

Taulukko 5. Voittajan tarkistaminen.

Käyttötapaus:	Voittajan tarkistaminen
Aktorit:	CPU
Ehdot:	CPU tai pelaaja on tehnyt onnistuneet voittosiirrot peliruutuihin.
Kuvaus:	Tutkitaan ruudukot ja tarkistetaan, onko ruudukkoihin muodostunut kolmen merkin suoria.
Poikkeukset:	Voittavia siirtoja ei ole tehty tai pystytty tekemään. Peli on loppunut.
Lopputulos:	Ruudukot on käyty läpi ja voittaja on ilmoitettu. Jos voittajaa ei löydy, peliä jatketaan.

Taulukko 6. Tasapelin tarkistaminen.

Käyttötapaus:	Tasapelin tarkistaminen
Aktorit:	CPU
Ehdot:	CPU tai pelaaja on tehnyt onnistuneet siirrot peliruutuihin
Kuvaus:	Voittaja ei ole selvillä, vaikka siirtoja on tehty 27.
Poikkeukset:	Siirtoa ei ole tehty tai pystytty tekemään. Peli on loppunut. Joko CPU tai pelaaja on voittanut pelin.
Lopputulos:	Ruudukot on käyty läpi ja voittaja ei ole selvillä 27 siirron jälkeen. Ilmoitetaan tasapelistä.

Taulukko 7. Helppo vaikeustaso.

Käyttötapaus:	Helppo vaikeustaso
Aktorit:	CPU
Ehdot:	Peli on aloitettu.
Kuvaus:	Vaikeustaso on aina helppo. Tietokone pelaa kuin aloittelija.
Poikkeukset:	Peliä ei ole aloitettu tai ohjelma ei ole käynnistynyt.
Lopputulos:	Tietokone pelaa kuin aloittelija, ja tekee välillä helppoja virheitä.

Taulukko 8. Siirto tilanteen mukaan.

Käyttötapaus:	Siirto tilanteen mukaan.
Aktorit:	CPU
Ehdot:	Siirtoja on jäljellä. CPU:n vuoro siirtää.
Kuvaus:	CPU tekee siirron tilanteen mukaan. Tietokoneella on ennalta määrätyt ratkaisut, joita se tekee pelaajan siirtojen mukaan.
Poikkeukset:	Siirtoja ei ole jäljellä. Vuoro on pelaajalla.
Lopputulos:	CPU on tehnyt siirron tilanteen mukaan. Vuoro siirtyy pelaajalle.

Taulukko 9. Tilanteen julistaminen

Käyttötapaus:	Tilanteen julistaminen.
Aktorit:	CPU
Ehdot:	Voittaja on selvillä tai on päädytty tasapeliin.
Kuvaus:	Voittajan jälkeen, ohjelma julistaa voittajan nimen. Jos pelissä on päädytty tasapeliin, niin tietokone ilmoittaa ruudulla tasapelistä.
Poikkeukset:	Siirtoja on jäljellä eli peli on kesken.
Lopputulos:	Tietokone ilmoittaa joko voittajan tai julistaa tasapelin.

5.3 Jätkänshakkipelin toiminnallisuus

Seuraavissa luvuissa esitellään tarkemmin kolmiulotteisen jätkänshakkipelin ohjelman rakennetta. Ensimmäiseksi käsitellään LCD-näytön alustusrutiineja, jotka asettavat graafisen LCD-näytön peliä varten toimintakuntoon. Samassa luvussa selvitetään näytölle piirtämisen toiminnallisuus. Näppäimistön osuudessa on selitetty matriisinäppäimistön toimintaa ohjelmatasolla: keskeytys, ruudukon valinta A-, B- tai C -kirjaimella ja merkin paikan valitseminen ruudukolla numeronäppäimillä. Pääohjelman osiossa kerrotaan, miten itse pelin toiminnallisuus on tehty.

5.3.1 LCD-näyttö

Aivan ensimmäiseksi täytyy graafinen LCD-näyttö alustaa, jotta ohjelmistoa voidaan käyttää. Näytön alustaminen tapahtuu kirjoittamalla alustustavut heksalukuina CONTROL- ja Data-rekistereiden osoitteisiin. Taulukossa 10 näkyvillä alustustavuilla alustetaan LCD-näyttö grafiikkatilaan: 256x64 eli ruudun leveys on 256 pikseliä ja korkeus 64 pikseliä. Control-rekisterin osoite kirjoitetaan ohjelmistoon muodossa ”0xC002”. Data-rekisterin osoite puolestaan kirjoitetaan muodossa ”0xC000”. Näyttö pitää välillä alustaa myös tekstitilaan. Tällöin Data-rekisteriin kirjoitettavat alustustavut ovat hieman poikkeavia. Nämä tavut on esitetty taulukossa 11.

Taulukko 10. Rekistereihin kirjoitettavat grafiikkatilan alustustavut.

Alustus	Control	Data
Mode Control	0x00	0x32
Character Pitch	0x01	0x77
Horizontal Count	0x02	0x1F
Multiplex Ration	0x03	0x3F
Cursor Position	0x04	0x07
Display Start Address (Low	0x08	0x00
Display Start Address (High	0x09	0x00
Cursor Address(Low)	0x0A	0x00
Cursor Address(High)	0x0B	0x00

Taulukko 11. Rekistereihin kirjoitettavat tekstitilan alustustavut.

Alustus	Control	Data
Mode Control	0x00	0x34
Character Pitch	0x01	0x75
Horizontal Count	0x02	0x27
Multiplex Ration	0x03	0x3F
Cursor Position	0x04	0x07
Display Start Address (Low	0x08	0x00
Display Start Address (Hig	0x09	0x00
Cursor Address(Low)	0x0A	0x00
Cursor Address(High)	0x0B	0x00

Alustus tapahtuu käytännössä siten, että tarkastellaan ensiksi ”BUSY”-lipun tilaa. BUSY-lippu on kahdeksas bitti. Kun lipun bitti on yksi osoitteessa 0xC003, niin silloin sen tila on varattu. Muissa tapauksissa tila on käytössä ja näytölle voidaan kirjoittaa tai piirtää. Kun kaikki yhdeksän heksalukua eli alustusmerkkiä on kirjoitettu rekistereiden osoitteisiin ja muut alustusrutiinit on tehty, niin LCD-näyttö on toimintavalmis. Muussa tapauksessa ohjelma joutuu aloittamaan alustuksen alusta.

Näytön alustus suoritetaan, jotta näytölle voidaan piirtää ruudukot ja ruudun kohdalle merkki ”X” tai ”O”. Näytölle piirtäminen aloitetaan alustamalla Control-rekisteri heksaluvulla 0x0C. Tällä saadaan ohjelma ymmärtämään, että sen pitää ohjata Data-linjoilta tuleva tieto suoraan näytölle. ”X:n” ja ”O:n” piirtäminen on toteutettu piirtämisrutiinien avulla. Merkkien piirtämiseen tarvittavat alustustavut löytyvät taulukosta 12. Tavut täytyy kirjoittaa kummankin merkin ”const char” -taulukkoon.

Taulukko 12. Merkkien “X” ja “O” piirtorutiinit.

Merkki "X"	**const char cross[]={0xC3, 0x66, 0x3C, 0x18, 0x18, 0x3C, 0x66, 0xC3};**
Merkki "O"	**const char circle[]={0x3C, 0x66, 0xC3, 0x81, 0x81, 0xC3, 0x66, 0x3C};**

LCD-näytön ohjelmistomoduuli LCD.c sisältää kaiken kaikkiaan LCD-näytön alustuksen (grafiikka- sekä tekstitila), piirtokursorin paikan asettamisen, näytön tyhjennyksen, merkkien piirtorutiinit, ruudukkojen piirtoaliohjelmat, tekstialiohjelmat, näytön tyhjennys- ja viivealiohjelmat.

5.3.2 Näppäimistön toiminnallisuus

Näppäimistön ohjelmisto on toteutettu siten, että näppäimen painaminen aiheuttaa keskeytyksen mikrokontrollerille, jolloin ohjelma siirtyy keskeytysaliohjelmaan. Keskeytys sallitaan kirjoittamalla erillinen käsky keskusyksikön IE-rekisteriin (Interrupt Enable). Rekisterin sisälöksi on kirjoitettava oikea tieto, jotta keskeytys tulee sallituksi. Oikea arvo kirjoitetaan rekisteriin joko bitti kerrallaan tai koko tavulle yhdellä kertaa. Näppäimistön keskeytyksenä käytetään ulkoista keskeytystä INT0. Ulkoinen keskeytys asetetaan toimimaan reunaherkästi, eli keskeytyksestä ei pyritä tekemään pysyvää. Keskeytysaliohjelma on valmis funktio, joka löytyy io51.h-kirjastosta.

Ensimmäiseksi pelaajan täytyy valita peliruudukko näppäimillä A, B tai C. Toiminto on toteutettu IF-lauseilla. A-näppäin tarkoittaa vasemmanpuoleista, B keskimmäistä ja C oikeanpuoleista alustaa. Seuraavaksi ohjelma jää odottamaan pelaajan peliruudun valintaa. Peliruutuja on jokaisella alustalla yhdeksän ja valinta tapahtuu painamalla näppäimistön numeronäppäintä. Pelaajan ruudukko- ja ruutuvalinta luetaan globaalimuuttujaan (PChoice-muuttuja), ja tämän jälkeen ohjelma hyppää keyb-aliohjelmaan. Aliohjelmassa pelaajan ruudukko- ja ruutuvalinta tulkitaan switch-case-rakenteella ja ohjelma jatkaa toimintaa sen mukaisesti.

5.3.3 Pääohjelma

Pääohjelma sisältää kaikki pelissä tarvittavat rutiinit. Kaikki tärkeimmät toiminnot on toteutettu erillisissä aliohjelmissa. Pääohjelman alussa näytölle tulostuu pelin tiedot ja ohje pelin käynnistämiseen. Kun pelaaja on painanut F-näppäintä, ohjelma piirtää kolme 3x3-peliruudukkoa. Ohjelma piirtää ruudukot LCD.c-ohjelmistomoduulin alustusrutiineja ja piirtoaliohjelmia hyväksi käyttäen. Pääohjelma jää odottamaan pelaajan siirtoa. Pääohjelma pyörii silmukassa 27 kertaa valinnat läpi, koska siirtoja on 27. Siirtojen päättyessä ohjelma julistaa tasapelin. Peli voi päättyä aikaisemminkin, jos ohjelma löytää kolmen merkin suoran, joko tietokoneelta tai pelaajalta. Voittaja tarkistetaan jokaisen siirron jälkeen. Peli loppuu voittajan tai tasapelin julistamisen jälkeen.

Pelin aloittaa aina pelaaja, ja pelimerkkinä toimii ”X”. Pelaajan täytyy ensiksi valita peliruudukko, mihin hän haluaa tehdä siirtonsa. Tämä tapahtuu painamalla A-, B- tai C-näppäintä. A on vasemmanpuoleisin, B seuraava ja C on viimeisin ruudukko. Seuraavaksi pelaajan täytyy valita ruutu numeronäppäimillä 1−9, mihin hän merkkinsä laittaa. Pelaajan siirron jälkeen vuoro siirtyy luonnollisesti tietokoneelle. ”O” on tietokoneen merkki.

Pääohjelmasta löytyy kokonaislukutaulukko (board[ ]), johon pelin siirrot rekisteröidään. Ohjelmassa pelaajan arvoksi on määritelty 1 ja tietokoneen arvoksi -1, jotta peli tunnistaa kummatkin pelaajat ja osaa kertoa voittajan. Tyhjän ruudun arvo on 0. Ohjelma tarkistaa jokaisen siirron jälkeen, onko voittaja selvillä. Jos siirtoja on tehty 27, eikä voittaja ole selvillä, ohjelma julistaa tasapelin. Peliä jatketaan niin kauan, että voittaja on selvillä tai siirtoja on tehty 27.

Voittajan etsiminen ja tasapelin julistaminen on toteutettu check_win-aliohjelmassa. Aliohjelmassa rivien tutkiminen hoidetaan IF-lauseilla. Jos tietokone löytää esimerkiksi ruudukoista kolme ykköstä peräkkäin (esim. board[0]==1 & board[1]==1 & board[2]==1), silloin pelaaja voittaa. Tietokone julistaa välittömästi voittajan näytöllä. Tietokone voittaa, jos ruudukoista löydetään kolme ”-1”-arvoa peräkkäin (esim. board[0]==-1 & board[1]==-1 & board[2]==-1). Jos aliohjelma kiertää silmukassa 27 kertaa läpi, eikä ohjelma löydä kolmen suoria, julistaa tietokone tasapelin.

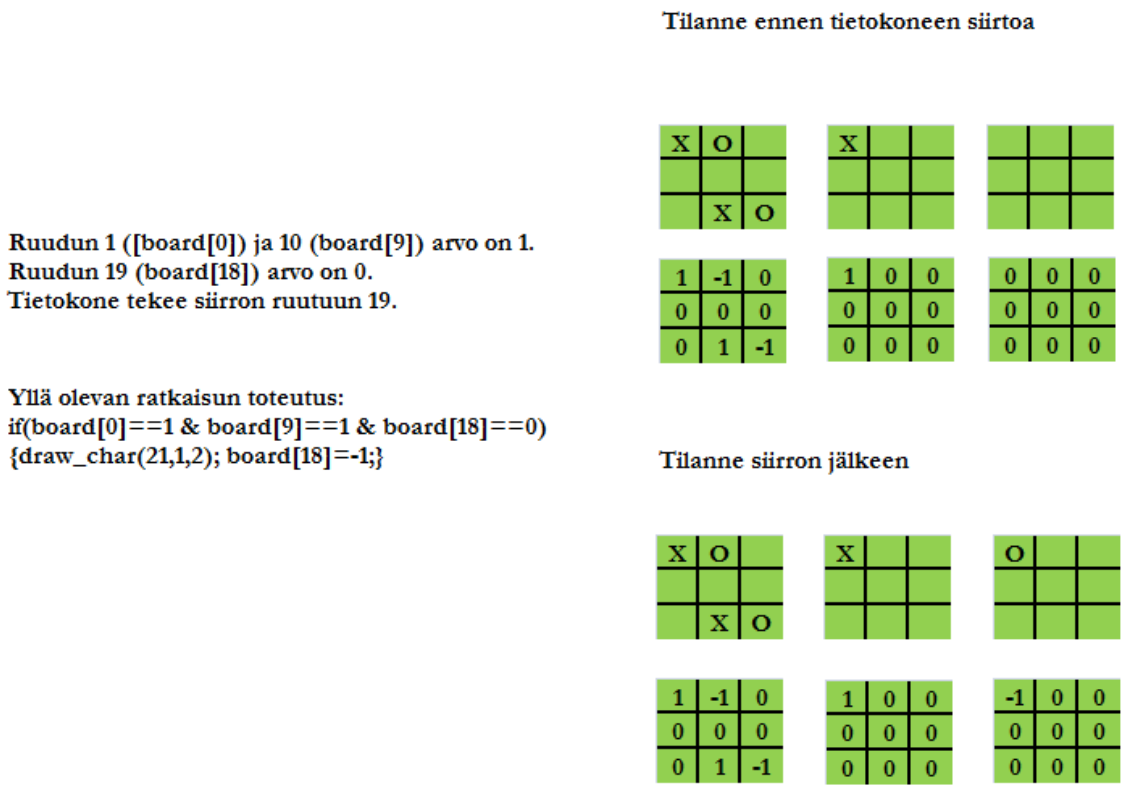
Ohjelmalla on yksi vaikeustaso. Tekoälyn vaikeustason voisi luonnehtia helpoksi, eli kokeneelle pelaajalle voittaminen ei ole vaikeaa. Aloittelijalla saattaa tulla vaikeuksia tietokoneen kanssa. Ohjelma käy jokaisen siirron jälkeen rivin läpi ja laskee rivien arvojen summat yhteen. Tietokoneen tekoäly on toteutettu aliohjelmassa computer_turn(). Aliohjelma on toteutettu IF-lauseilla. Lauseiden avulla tietokoneen siirrot on määritelty. Siirroissa on käytetty apuna kokonaislukutaulukkoa. Tietokone pyrkii ensisijaisesti estämään pelaajan kolmen suorat. Seuraavassa luvussa on kerrottu tarkemmin tekoälyn toiminnasta. Liitteessä 1 on esitelty pelin toimintaa vuokaavion avulla.

5.3.4 Tekoäly

Tekoälyosuus rakentuu yhdestä aliohjelmasta. Aliohjelma on toteutettu IF-lauseilla. Jokaiselle ruudukolle on varattu yksi kokonaislukutaulukko. Kokonaislukutaululla on arvona joko 1, 0 tai -1. Ensimmäinen ruutu on taulukossa ”board[0]”, seuraava on ”board[1]” ja jne. Taulukoita on yhteensä 27, kuten ruudukoitakin. Aliohjelmassa käydään jokainen määritelty rivi läpi IF-lauseilla, ja tietokone tekee ratkaisut ja siirrot niiden mukaisesti.

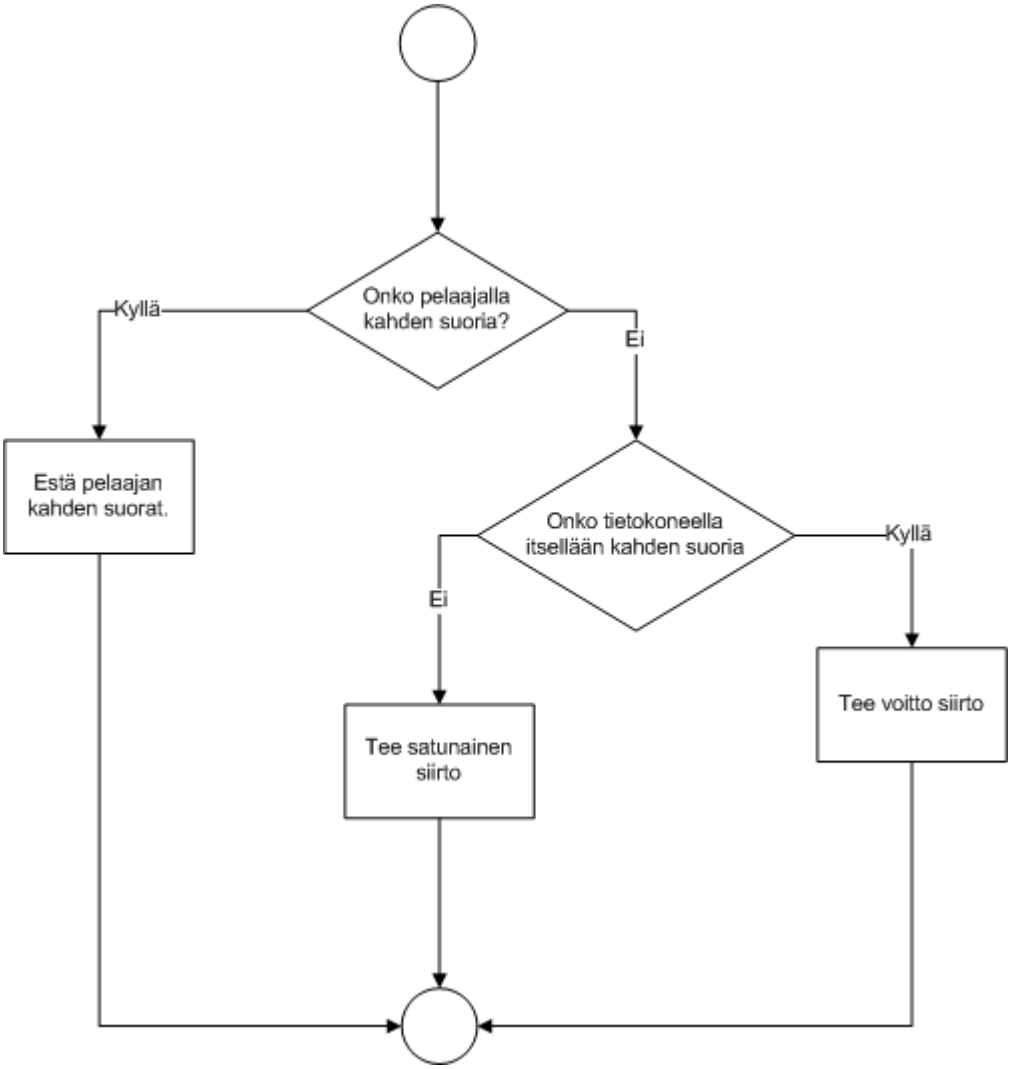
Peli perustuu pisteytykseen. Pelaajan siirroille annetaan arvoksi 1, ja tietokoneelle arvoksi tulee -1. Näiden lukuarvojen perusteella tietokone voi tutkia, millainen pelitilanne on kyseessä ja tehdä siirtonsa sen mukaisesti. Numeroarvojen perusteella ohjelma tietää, pystyykö kyseiseen ruutuun tekemään siirron. Ruudun arvon ollessa 0 siirto on mahdollinen, muussa tapauksessa siirto ei onnistu. Ohjelma pystyy kokonaislukutaulukon arvojen perusteella valitsemaan oikean IF-lauseen, ja tietokone tekee tämän avulla ratkaisun ja siirtonsa. Tietokone pystyy myös estämään pelaajan kolmen ja kahden suorat tällä samalla menetelmällä. Kuvassa 10 on selvennetty äskeistä menetelmää.

Kaikki ruudukot käydään läpi ja ohjelma tarkistaa löytyykö ruudukoista kolmen suoria pysty-, vaaka-, vino- tai syvyyssuunnassa. Ohjelma pyrkii estämään tai tekemään voittavan siirron tilanteen mukaan. Ohjelma ei löydä kuvan 10 esimerkkitilanteesta yhtään -2 riviä eli sillä ei ole mahdollisuutta tehdä voittavaa siirtoa. Esimerkkitilanteessa ohjelma huomaa, että kahden kokonaislukutaulukon yhtälöksi tulee (ks. kuva 10): ”(-1)+(-1)+0=2”. Tietokone ymmärtää, että sen on estettävä pelaajan voitto, koska yhtälön arvoksi tulee kaksi. Alla olevan kuvan 10 esimerkissä on selvitetty, miten tietokone lukee peliä.



Kuva 10. Esimerkkitilanne tietokoneen ajattelutavasta.

Tietokone ohjelmoidaan estämään pelaajan kahden ja kolmen suorat ensisijaisesti. Tekoäly pystyy rakentamaan kahden suoria satunnaisesti, eli suorat syntyvät lähinnä pelaajan virheistä. Ohjelmoidaan tietokone valitsemaan sopivia siirtoja IF-lauseiden avulla. Kuvassa 11 on esitetty tekoälyn toiminta vuokaavion avulla.

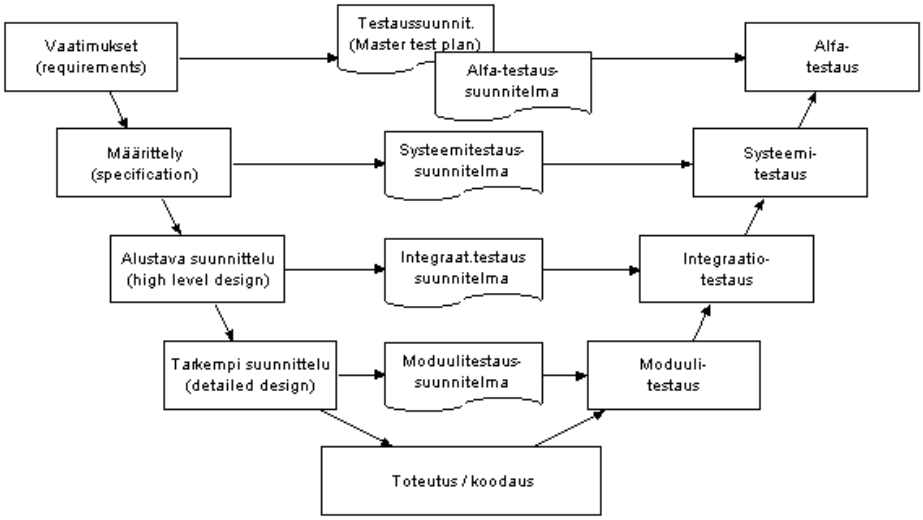


Kuva 11. Tekoälyn toimintalogiikka.

5.4 Testaus

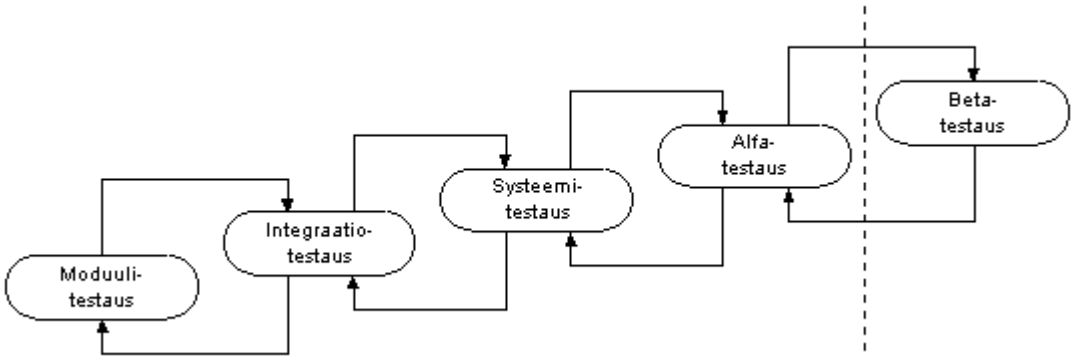
5.4.1 Testausmenetelmät

Lukuun ottamatta joitakin pieniä ja yksinkertaisia ohjelmia, testaus tulisi käsittää kokonaisena prosessina ohjelmistokehityksen aikana. Testausprosessin suhdetta ohjelmistoprosessiin kuvataan usein seuraavanlaisella V-mallilla (ks. kuva 12). [6.]



Kuva 12. Testausprosessin V-malli.[6.]

Testausprosessit jaetaan pienempiin osiin, jotta niitä olisi helpompi hallita. Näitä osia kutsutaan tasoiksi (eng. testing levels). Kuvassa 13 on näkyvillä testausprosessin tasot. Jos testattava kohde ei läpäise jotakin testivaihetta, se palautetaan tasolle, jolla virhe on tapahtunut tai jolla virheen uskotaan olevan. Tämän jälkeen virheet ja ristiriidat tarvittaessa paikannetaan, korjataan ja kohde testataan uudelleen. Usein puhutaan regressiotestauksesta, jolla tarkoitetaan, että kaikki virhettä edeltäneet testit suoritetaan uudelleen varmentaen, etteivät tehdyt korjaukset ja muutokset ole synnyttäneet uusia virheitä alemmille tasoille [6.]. Seuraavissa kappaleissa on käyty yksityiskohtaisesti ne testauksen tasot, joita on tarvittu kolmiulotteisen jätkänshakkipelin ohjelmiston testauksessa. Alfa- ja beta-testausta ei tarvittu, koska ne koskevat lähinnä asiakasta tai asiakkaita. Omassa työssäni testauksessa ei tarvitse edetä alfa- ja beta-testauksen tasolle. Tuotteelle, jolla oletetaan olevan markkina-arvoa, suoritetaan alfa- sekä beta-testaus.



Kuva 13. Testauksen tasot(ks. kuva 12).[6.]

Moduulitestaus (yksikkötestaus, unit testing) käsittää pienimpien ohjelmayksiköiden, esimerkiksi funktioiden, ja niistä muodostuvien pienten kokoelmien eli moduulien, testauksen. Moduulitestauksen tavoitteena on löytää selvien virheiden lisäksi ristiriitoja yksiköiden ja moduulien määrittelyjen ja toiminnan välillä sekä korjata ne.[6.]

Integraatiotestauksella tarkoitetaan vaihetta, jossa pienemmät osat kootaan yhteen ja testataan, kunnes koko systeemi on saatu kasaan. Integraatiotestauksen tarkoituksena on varmistaa, että kasatut osat toimivat määritellysti keskenään, sekä löytää ja korjata mahdollisimman paljon yhteen soveltumattomia moduuleita.[6.]

Systeemitestauksen päätavoite on tutkia, täyttääkö valmis systeemi (esim. ohjelmisto) sen määrittelyn asettamat vaatimukset. Toisaalta tavoitteena voidaan taas pitää sitä, että löydetään ja korjataan mahdollisimman paljon ristiriitoja systeemin ja sen määrittelyn väliltä. Joskus voi olla lisäksi tarpeellista testata esimerkiksi valmiin ohjelman suorituskykyä, rasituskestävyyttä, turvallisuutta ja toimivuutta verkossa.[6.]

Ohjelmistolle tehtiin myös staattinen ja dynaaminen analyysi. Seuraavissa kappaleissa on puolestaan käyty läpi staattinen ja dynaaminen analyysi. Dynaamisesta analyysista on kerrottu sen strategiatavoista, black box - ja white box -testaus, koska ne ovat olennaisia menetelmiä testilaitteiston testauksessa.

Staattisessa analyysissä etsitään virheitä ja puutteellisuuksia tutkittavasta kohteesta ilman ohjelmakoodin suorittamista. Seuraavassa on lueteltu esimerkkejä, minkä tyyppisiä virheitä staattisessa analyysissä voidaan paljastaa:

– muuttujien väärinkäyttö

– osoittimien ja indeksien väärinkäyttö

– parametrien väärinkäyttö

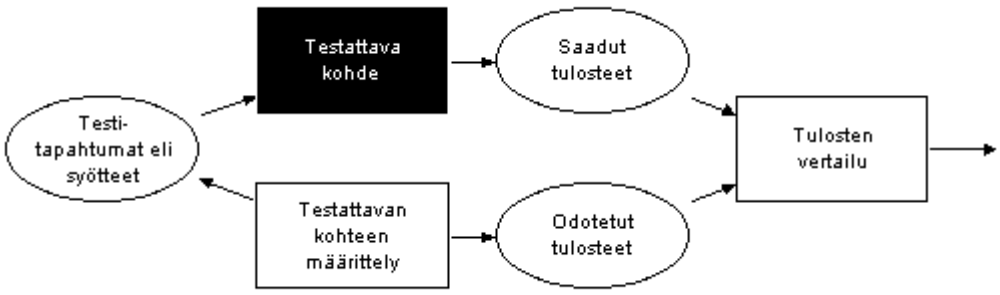
– kontrollivirheet (esim. saavuttamaton koodi)

– funktioiden väärinkäyttö

Suurin osa edellä mainituista virheistä käy ilmi jo kääntämisen aikana. Kääntäjä on jo valmiiksi yksi tärkeä työkalu testauksen kannalta.[6.]

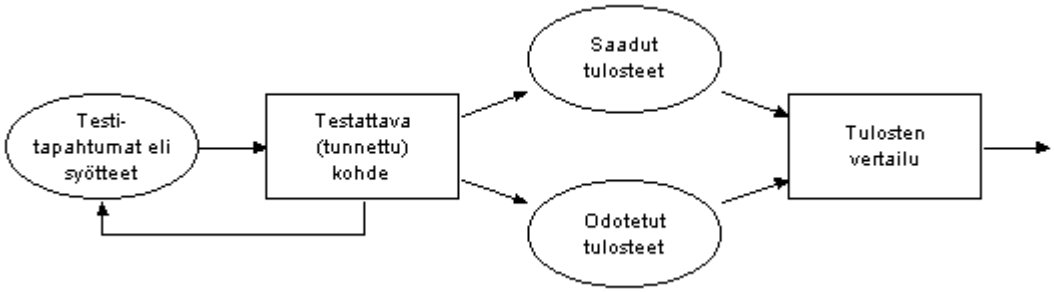
Dynaamisessa analyysissä keskitytään ohjelman ja koodin käyttäytymiseen suorituksen aikana. Testaus edellyttää siis jonkinlaista toimivaa ohjelmaa tai ympäristön, missä esimerkiksi yksittäisiä funktioita voidaan suorittaa. Tällä tavoin saadaan testattua ominaisuuksia, mitkä staattisessa analyysissä olisivat mahdottomia tai ainakin vaikeita testata.[6.]

Black box -testaus perustuu esimerkiksi testattavan ohjelman input-output käyttäytymiseen. Black box -testauksessa ei välitetä testattavan kohteen rakenteesta tai sisällöstä, vaan tutkimuksen kohteena ovat tulosteet (output) erilaisilla syötearvoilla (input). Testaajalle kohde on tuntematon "musta laatikko", black box (ks. kuva 14). Testattavan kohteen oikeellisuutta tarkastellaan vertaamalla saatuja tulosteita haluttuihin tai odotettuihin tulosteisiin. Testitapaukset johdetaan aina kohteen määrittelyn perusteella.[6.]



Kuva 14. Black box -testaus.[6.]

White box -testauksessa testitapaukset johdetaan kohteen, esimerkiksi koko ohjelman sisäisestä rakenteesta ja logiikasta (ks. kuva 15), toisin kuin black box -testauksessa. Kaikki kohteen, esimerkiksi ohjelman, testitapaukset pyritään valitsemaan siten, että kaikki kohteen haarat ja ohjelmapolut tulisi käytyä läpi.[6.]



Kuva 15. White Box -testaus.[6.]

5.4.2 Testauksen toteutus

Ohjelma on testattu kahdessa osassa. Ensimmäiseksi testattiin sulautetun järjestelmän ohjelmat, kuten esimerkiksi näppäimistön ja näytön toiminnallisuus. Testausympäristönä käytettiin testilaitteen kanssa PC-tietokonetta, IAR ANSI C-kääntäjää ja EMUL51-emulaattoria. Toiseksi testattiin ohjelman PC-osuus, joka toimii ainoastaan PC-ympäristössä (IAR Embedded Workbench Kickstart Edition 4.0 -versio). PC-osuus sisältää pelin toiminnallisuuden. Kotikannettavalla oli helpompi testata itse pelin toiminta ja siksi ohjelmiston testaus piti jakaa kahteen ympäristöön. Testauksen jälkeen PC-versio on tehty yhteensopivaksi käytetyn järjestelmän kanssa.

Ensimmäiseksi testattiin sulautetun järjestelmän ohjelmiston toiminnallisuus. Testausmenetelmänä on käytetty moduuli- ja järjestelmätestausta. Ohjelmistolle on myös tehty staattinen analyysi, eli ohjelmisto on ajettu kääntäjästä läpi.

PC-versiolle on tehty omat testitapaukset. Koko toiminnallisuus on ajettu useaan kertaan läpi kääntäjästä eli ohjelmalle on tehty staattinen analyysi. Ohjelmiston aliohjelmat on testattu käyttämällä dynaamisen analyysin black box - ja white box -testausta. Lopuksi on tehty vielä järjestelmätestaus.

Testauksen seuraavassa vaiheessa molemmat ohjelmistomoduulit yhdistettiin kokonaisuudeksi ja syntyneelle ohjelmistolle ajettiin integraatiotestaus. Sovellukselle tehtiin lopuksi käyttöliittymä- ja järjestelmätestaus.

6 LOPPUTULOKSET JA ANALYYSIT

Aivan alkuperäisenä ideana oli tehdä läpinäkyvä kuutio, jossa peliä olisi voitu pelata. Luovuin tästä ideasta jo hyvissä ajoin. Totesin sen olevan hankalaa toteuttaa C-kielellä ja päädyin toiseen ratkaisuun. Ohjelmoin ohjelman piirtämään näytölle kolme vierekkäin olevaa peliruudukkoa läpinäkyvän kuution sijasta. Käytin apuna Kari Niskasen lähdekoodia.

Tekoäly on toteutettu siten, että tietokone pyrkii ensisijaisesti estämään pelaajan kolmen suorat ja etsii samalla omia kahden suoria. Tietokoneelle on määritetty tietyt paikat ruudukolla IF-lauseiden avulla. Siirrot eivät ole sattumanvaraisia. Ohjelma tarkistaa jokaisen IF-lauseen ja valitsee tilanteeseen sopivan siirron. IF-lauseisiin on kirjoitettu ehtoja kokonaislukutaulujen avulla. Jokaisella peliruudulla on oma kokonaislukutaulunsa. Taulut sisältävät kolme arvoa, joiden perusteella tietokone tietää onko ruutu tyhjä vai onko ruudussa ”X”- tai ”O”-merkki. Pelaajan arvo on 1, tietokoneen -1 ja tyhjän ruudun arvo 0. Edellä mainituilla arvoilla ohjelma tunnistaa ruudukon. Tämä ei takaa sitä, että peli päättyisi aina tasan tai tietokoneen voittoon. Tehtävän annossa tämä ei ollut kuitenkaan tavoitteena, kuten Kari Niskasen työssä. Ohjelman tarkoituksena on vain luoda toimiva tekoäly, jotta peliä pystytään pelaamaan. Tekoälyllä ei ole vaikeustasoja eikä se ole itseoppiva, mutta se tarjoaa silti pelaajalle hieman vastusta.

Kolmiulotteisuus eli tässä tapauksessa kolme tasoa tuo pelille oman mausteen. Siirron voi tehdä kolmelle eri tasolle ja voiton voi saavuttaa syvyyssuunnassa. Voiton saavuttaa, jos jompikumpi pelaajista saa esimerkiksi merkin jokaisen pelialustan keskelle. Tämä mahdollisuus lisää huomattavasti pelin mielenkiintoa ja vaikeuttaa pelaajan pelaamista.

Lopputulos on tyydyttävä. Ohjelma toimii ja sillä pystyy pelaamaan jätkänshakkia. Graafinen puoli ei oikein onnistunut, vaan kolmiulotteisuus on korvattu kolmella vierekkäin piirretyllä pelialustalla. Kolme pelialustaa riittää tässä tapauksessa, jotta peli saadaan näyttämään ”kolmiulotteiselta”. Tämä ratkaisu oli paras mahdollinen, jonka pystyin toteuttamaan.

7 YHTEENVETO

Työn tarkoituksena oli suunnitella ja ohjelmoida C-kielellä kolmiulotteinen jätkänshakkipeli valmiiseen testilaitteistoon. Työ tehtiin Kajaanin ammattikorkeakoululle. Aihe liittyi läheisesti Kari Niskasen insinöörityöhön, jonka aiheena oli jätkänshakkipeliautomaatti. Oma työni oli lähes samanlainen testilaitteiston osalta. Käytin hyväksi Kari Niskasen ideoita, tietenkin soveltaen niitä. Niskasen työ keskittyi lähinnä tekoälyyn ja sen ”itseoppivaisuuteen”, kun taas omassa työssäni lähtökohtana oli kolmiulotteisuus. Omassa työssäni riitti, että ohjelmalla pystytään pelaamaan jätkänshakkia ja tietokoneella on oma tekoäly.

Työ oli haastava, koska tietoni C-kielestä ja sulautetun järjestelmän ohjelmoinnista olivat päässeet unohtumaan. Luovuin ideasta toteuttaa kolmiulotteisuus läpinäkyvässä kuutiossa jo hyvissä ajoin aikataulun ja sen vaikeuden takia. C-kieltä ei ole oikein luotu 3D-graafikan ohjelmointiin. Päätin toteuttaa kolmiulotteisuuden siten, että näytöllä näkyisi kolme pelialustaa vierekkäin.

Pelin grafiikan toteuttaminen onnistui soveltamalla Kari Niskasen käyttämään koodia, jotta pystyin luomaan omalle sovellukselle käyttöliittymän ja grafiikan [1.]. Itse pelin toiminnallisuus oli kohtalaisen helppo rakentaa omalla kannettavalla. Tekoälyn luominen onnistui ilman suurempia ongelmia. Toiminnallisuuden ohjelmointi testilaitteistolle tuotti myös hankaluuksia, koska PC:llä luotu pääohjelma piti saada yhteensopivaksi grafiikkaohjelmamoduulin kanssa, jotta ohjelma toimisi testilaitteistossa. Myös testilaitteisto oli minulle outo, koska labratietoni olivat päässeet unohtumaan. Varsinkin tietoni LCD-näytöstä olivat aluksi heikot. Laitteistoon tutustumiseen kului normaalia enemmän aikaa.

Tehtävän annossa pääkriteerinä oli kolmiulotteisuus ja tekoäly oli toissijainen tehtävä. Kolmiulotteisuuden toteutus jäi hieman vajaaksi sen vaikeuden takia. Tekoäly toimii moitteettomasti eli tietokone tarjoaa pelaajalle kyllä vastusta jonkin verran. Pääasia oli kumminkin, että sovelluksessa pystytään pelaamaan peliä. Pelissä on vain yksi vaikeustaso ja pelin päättyessä se alkaa alusta.

Lopputuloksena oli kolmiulotteinen jätkänshakkipeli, jota pystytään pelaamaan tietyillä rajoituksilla. Ohjelmaa voidaan käyttää, kun esitellään esimerkiksi tietotekniikan koulutusalan mahdollisuuksia.

LÄHTEET

1. Kari Niskanen, Jätkänshakkipeliautomaatti insinöörityö, 2006, luettu viimeksi 8.4.2008, [WWW-dokumentti],

   https://kaktus.kajak.fi/Teli/TTI2SKariN.pdf

2. Jack Bottermans, Tony Burret, Pieter Van Delft, Carla Van Splunteren, Suuri pelikirja, Kustannus-Mäkelä Oy, 1990, 103 s., ISBN 951-873-281-7

3. Kari Silpiö, C-kieli, ATK-instituutti, 1995, 1-6 s., ISBN 951-37-1041-6

4. Pekka Rantala, Mikrotietotekniikka tietotekniikka osa 2, Tietokotka, 2001, 17, 86–87 s., ISBN-951–559-258-5

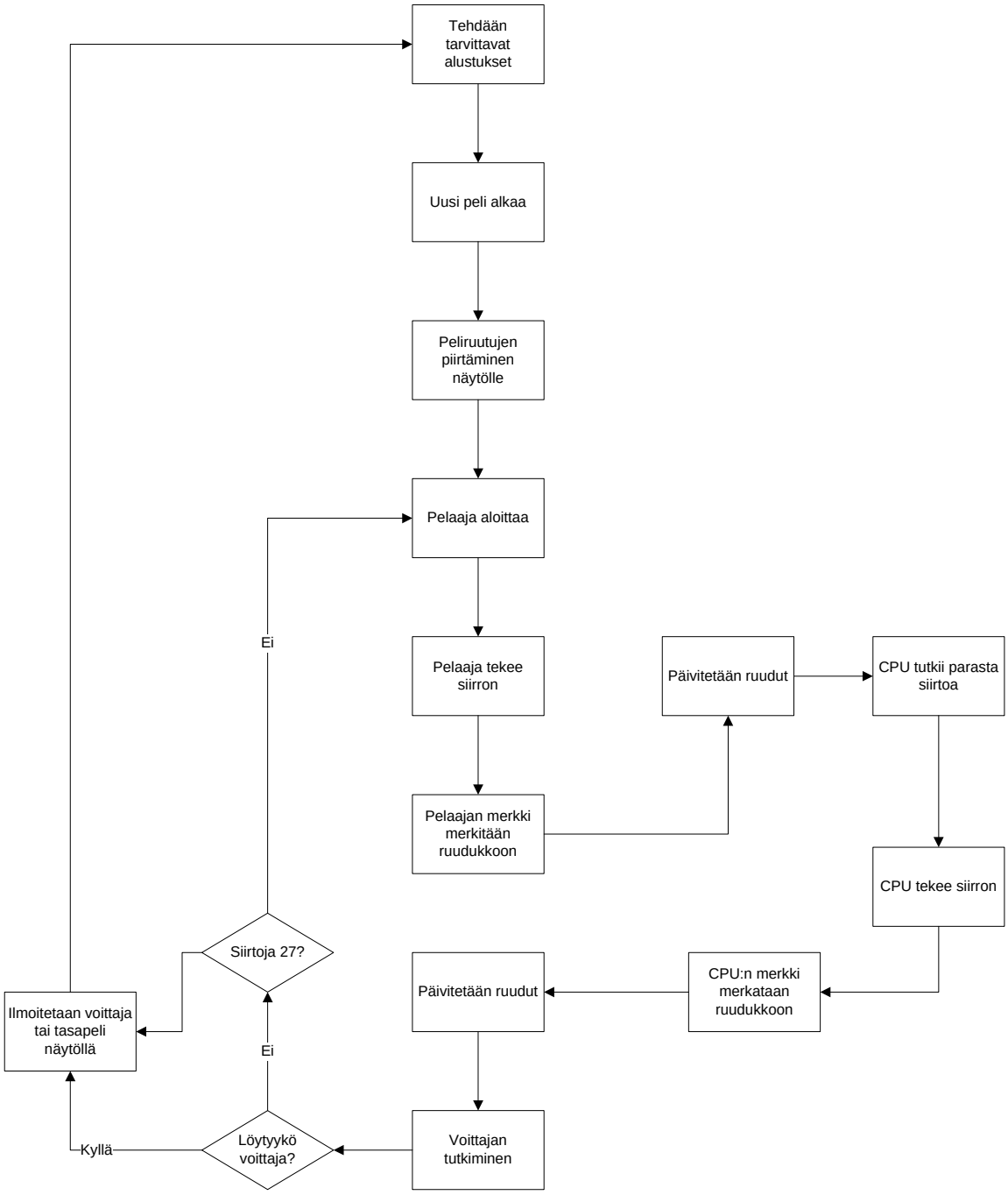
5. Hitachi, HD61830/HD61830 datasheet, luettu viimeksi 23.3.2008, [WWW-dokumentti],

   http://www.eio.com/hd61830.pdf

6. Tuomas Kautto, Ohjelmistotestaus ja siinä käytettävät työkalut, luettu 17.3.2008, [HTML-dokumentti],

   http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/

LIITTEIDEN LUETTELO


Tehdään tarvittavat alustukset
Uusi peli alkaa
Peliruutujen piirtäminen näytölle
Pelaaja aloittaa
Pelaaja tekee siirron
Päivitetään ruudut
CPU tutkii parasta siirtoa
Pelaajan merkki merkitään ruudukkoon
CPU tekee siirron
CPU:n merkki merkataan ruudukkoon
Päivitetään ruudut
Voittajan tutkiminen
Löytyykö voittaja?
Kyllä
Ei
Ilmoitetaan voittaja tai tasapeli näytöllä
Siirtoja 27?
Ei