Lappeenrannan teknillinen yliopisto
Konetekniikan osasto
Mekatroniikan ja virtuaalisuunnittelun laboratorio 20.2.2008

KANDIDAATINTYÖ

Haptiikkalaitteen hankinta ja käyttöönotto

0238208 Pekka Pellinen

Sisällysluettelo

1 Johdanto

Tehtävänä on hankkia haptiikkalaite, jolla saadaan tuntopalaute virtuaalitodellisuudesta. Lisäksi suoritetaan laitteen asennus ja tehdään yksinkertainen haptiikkaohjelma, jossa laite kiinnitetään jouseen virtuaalitodellisuudessa.

Haptiikka on vuorovaikutusta tuntoaistin kautta. Tämän avulla voidaan viedä laitteiden suunnittelu paljon pidemmälle tietokoneiden avulla kuin nykyisin pelkällä paperiversiolla tai 3D-mallilla. Yksistään haptiikkalaitteen avulla ei kuitenkaan saada aikaan toimivaa järjestelmää, vaan lisäksi tarvitaan myös visuaalinen käyttöliittymä.

Suunnittelussa täytyy tuntopalautteen kuitenkin olla mahdollisimman realistinen, samoin kuin visualisoinnin. Tämä antaa mahdollisuuden luoda hyvin lähellä realistista maailmaa olevan mallin, jonka toimivuutta ihminen voi kokeilla.

Tällaisesta mahdollisuudesta tulee olemaan suunnittelulle suuri hyöty kustannuksissa. Tuotteesta tuleva palaute saadaan aikaisemmassa vaiheessa suunnitteluprosessia, koska ei tarvitsisi rakentaa prototyyppiä ja malleihin tehtävät muutokset tehtäisiin virtuaalitodellisuudessa oleviin malleihin.

2 Käytetyt menetelmät

Laitteita on olemassa erilaisia, mutta kaikissa on tarkoitus tuottaa realistinen tuntopalaute virtuaalitodellisuudesta. Markkinoilla olevilla laitteilla tuntopalaute luodaan joko voimatakaisinkytennällä tai värinällä.

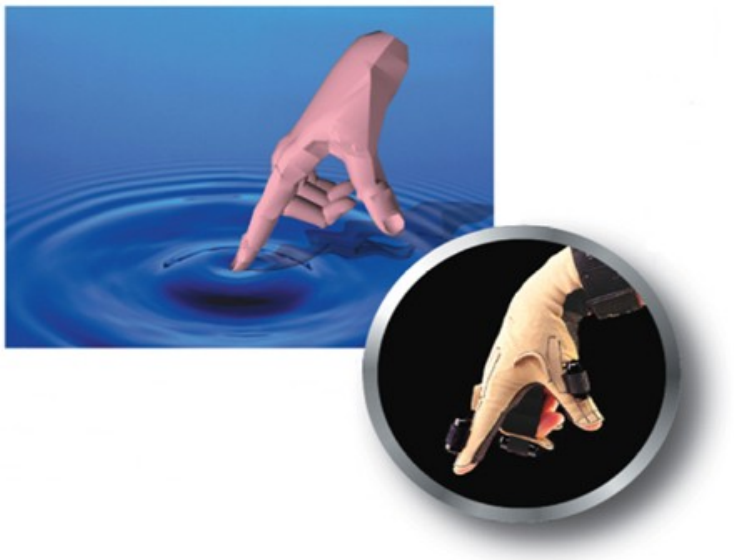
Voimatakaisinkytkentä tapahtuu jonkin tapaisella mekaanisella välityksellä. Yleisimpiä tällaisia laitteita markkinoilla ovat käsivarsi- tai rinnakkaisrobotti periaatteella toteutetut laitteet

Kuva 1: Käsivarsi periaatteella toteutettu haptiikkalaite (Inition 2006).

Kuva 2: Rinnakkaisrakneteella toteutettu haptiikkalaite (Inition 2005).

Voiman tuotossa rinnakkaisrobotti periaatteella saadaan aikaan suurempi voima kuin robottikäsivarsi periaatteella toimivasta. Työtila on kuitenkin suurempi käsivarsi periaatteella toimivassa kuin rinnakkaisrakenteisessa. Työtilojen muodoissa on myös eroa. Käsivarsi periaatteella toimivassa on kuution muotoinen työtila ja rinnakkaisrakenteisessa on puolipallon muotoinen työtila.

Datahanskan sallima työtila riippuu onko laiteen paikkatiedon lukeminen langaton vai mekaaninen yhteys. Langattoman työtila on suurempi, minkä rajoite on laitteen lähetys teho. Alla kuva datahanskasta.



Kuva 3: Datahanska jossa vibraattorit sormien päissä (Inition 2008a).

Vapausasteet, jotka laitteet vähintään tunnistavat, ovat translaatio x-, y- ja z-akselin suhteen. Osasta laitteista saadaan myös kierto edellä mainittujen akseleitten suhteen. Voimatakaisinkytkentä on myös mahdollista saadaa kaikkien akseleiden suhteen.

Parhaimman tuntopalautteen antavat laitteet ovat todella monimutkaisia laitteita. Näissä on yhdistetty käsivarsirobotti ja datahanska, jossa on oma mekanismi voimatakaisinkytkennälle. Tällaisella laitteella saadaan aikaan jokaisessa sormessa kosketuksesta syntyvä voimatakaisinkytkentä. Tällaisesta laitteesta on kuva alla.

Kuva 4: Datahanskan ja käsivarsirobotin yhdistelmä (Inition 2008b).

2.1 Laiteen hankinta

Ensimmäinen haptikkalaite tulee olla yksinkertainen, jotta haptisenlaitteen käytön opettelu ei olisi liian vaikeaa. Eli ensimmäinen laite jolla harjoitellaan tuntopalautteen käyttämistä suunnittelun apuna on joko kuvassa 1 tai 2 esitetyn laitteen kaltainen.

Laitteelta vaaditaan seuraavia ominaisuuksia riittävä työtila, jolloin sillä voidaan käyttää erilaisten laitteiden kojetauluja ja muita hallintalaitteita. Voimantuoton täytyy olla riittävä, etteivät kovat pinnat tunnu kuin painaisi ilmapalloa Tämä saavutetaan riittävällä maksimivoimalla jolloin törmäys tasoon saadaan tuntumaan realistiselta Paikkatieto täytyy saada kuuden vapausasteen suhteen, jolloin voidaan kääntää kappaleita tai kameraa. Laitteessa täytyy myös olla nappi, jolla voidaan joko tarrata jostakin kiinni tai siirtää kameran paikkaa virtuaalitodellisuudessa. Alla ovat listattu tarkemmin laitteelta vaadittavat ominaisuudet:

- työtila vähintään 300x200x170 mm (korkeus x leveys x syvyys)
- voiman tuotto vähintään 30 N ja jatkuva voima vähintään 5 N

- tunnistaa paikan 6 vapausasteen suhteen (translaatio ja rotaatio x-, y- ja z-akselin suhteen)
- voimatakaisinkytkentä vähintään kolmen vapausasteen suhteen (translaatio x-, y- ja z-akselin suhteen).
- nappi.

Hankittavan ohjelmistolta vaadittavat ominaisuudet:

- tukee käyttöjärjestelmää Windows XP.

Laite, joka täyttää edellä olevat vaatimukset on PHANTOM® PREMIUM 1.5 HIGH FORCE, joka on käsivarsi periaatteella toteutettu haptiikkalaite. Alla taulukossa 1 ovat laitteen tarkemmat tiedot.

Taulukko 1: Hankittavan laitteen ominaisuudet (User's Guide 2004).

PHANTOM® PREMIUM 1.5 HIGH FORCE	
Työtila [(leveys)x(korkeus)x(syvyys) mm]	381x267x191 mm
Maksimi voima [N]	37,5 N
Jatkuva voima [N]	6,2 N
Paikan tunnistaminen, vapausasteet	6 vapausastetta
Voiman tuottaminen, vapausasteet	3 vapausastetta (translaatiot)

Laitteen ohjelmisto on OpenHaptics™ Academic Edition, jonka laitteistovaatimukset, taulukossa 1 olevan laitteen kanssa, ovat alla taulukossa 2.

Taulukko 2: Hankittavan ohjelmiston ominaisuudet (PROGRAMMER'S GUIDE 2004).

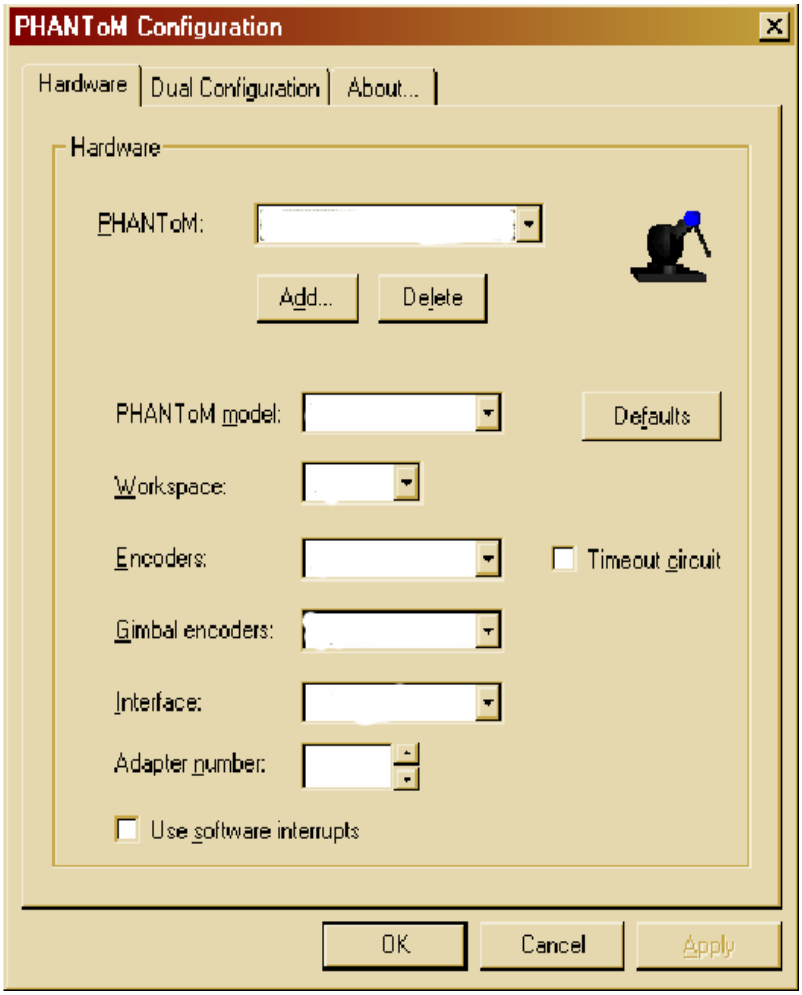
OpenHaptics™ Academic Edition	
Laitteistovaatimukset	Intel 4 tai tehokkaampi 100 MB kovalevytilaa 256 MB RAM-muistia Rinnakkaisportti
Käyttöjärjestelmät	Windows: • 2000 • XP Linuxin jakelut (ytimen versio 2.4.x): • Red Hat(R) • Fedora(TM) Core 1, 2, 3, 4 • SuSE 9.1, 9.2, 9.3
Kääntäjä	Windows: • Microsoft® Visual C++ 6.0 tai uudempi • MicroSoft Visual Studio .Net 2003 Linux: • gcc 3.x

Ohjelmiston lisenssi on ilmainen koulutus ja tutkimus käyttöön ja lisenssin voimassaoloaika on yksi vuosi. Tämän jälkeen lisenssi täytyy uusia SensAble:n Developer Support Center:issä, joka löytyy osoitteesta http://dsc.sensable.com/.

2.2 Laitteen käyttöönotto

2.2.1 Ajureiden asennus

Ennen ajureiden asennusta täytyy poistaa laitteen vanhat ajurit, jos ne ovat asennettuna. Tämä tapahtuu ”Lisää tai poista” -sovellus kuvakkeesta, joka sijaitsee ”Ohjauspaneelissa”. Tämä jälkeen asetetaan CD-levy CD-asemaan ja ajurit asennetaan asennusohjelmalla, joka sijaitsee kansiossa ”Phantom Device Drivers”. Asennusohjelma aukaisee valikon, joka on esitetty alla kuvassa 6 (User's Guide 2004).



Kuva 5: Asennusohjelman valikko (User's Guide 2004).

Alasvetovalikkoon asetetaan seuraavat seuraavat asetukset alasvetovalikoista, jotka on esitetty alla taulukossa 3.

Taulukko 3: Asennusohjelman asetukset (User's Guide 2004).

Valikko	Asetus
PHANToM	Default PHANToM
PHANToM model	Premium A
Workspace	1.5
Encoders	High Resolution
Gimbal encoders	High Resolution (jos asennettuna)
Interface	Paraller port

Kohdassa ”Timeout circuit” ei pidä olla merkkiä, ja jos tietokoneeseen on asennettu erillinen rinnakaisporttikortti PCI-väylään täytyy laittaa rasti kohtaan ”Use software interrupts”, jos normaalit keskeytykset eivät toimi (User's Guide 2004).

Kun asetukset on laitettu, käynnistetään tietokone uudelleen. Ennen uudelleen käynnistämistä joudutaan tarkastamaan laitteen ohjekirjasta tarvitseeko tietokoneen BIOS:in asetuksia muuttaa, koska laite asennetaan tietokoneeseen, jossa on Windows-käyttöjärjestelmä (User's Guide 2004).

Laitteen ohjekirjan kappaleessa 4, Computer Setup for Windows, sivulla 11 on taulukko, jossa on listattu laitteita joihin tarvitsee tehdä muutoksia. Muutokset koskevat rinnakkaisportin tiedonsiirtoprotokollaa, eli käytetäänkö ECP vai EEP protokollaa. Protokollat eroavat vain hieman toisistaan, alla on kerrottu hieman protokollista:

- EPP (*Enhanced Parallel Port*) on kaksisuuntainen rajapinta, joka on suunniteltu laitteille, jotka siirtävät suuria määriä tietoa. Teoreettinen siirtonopeus on 1,5 Mbit/s (IEEE 1284: Parallel Ports 2002).

- ECP (*Extended Capability Port*) on muuten samanlainen kuin EEP, mutta ECP:ssä on DMA (*Direct Memory Access*), jonka avulla voidaan tietoa siirtää suoraan muistista tai muistiin käyttämättä prosessoria. Teoreettinen siirtonopeus on 2,5 Mbit/s (IEEE 1284: Parallel Ports 2002).

Sovelluskehityksessä käytettävää tietokonetta ei löydy listasta ja protokollan oikeellisuus tehdään kokeilemalla. Tiedonsiirto toimii, kun protokollaksi asetetaan ECP+EEP.

Uudelleen käynnistämisen jälkeen laite tunnistetaan ja tämän jälkeen voidaan suorittaa Phantom Test -ohjelma, jolla voidaan kalibroida laite, lukea antureita, asettaa moottoreille voima ja kokeilla laitteen toimintaa.

2.2.2 OpenHaptic (TM) Toolkit educational:in asennus

Seuraavana asennetaan OpenHaptics (TM) Toolkit educational. Uusimman ohjelmaversion saa ladattua SensAble Developer Support Center:stä. Ladatut tiedostot poltetaan CD-levylle ja asennetaan CD-levyltä. Tämä tapahtuu suorittamalla asennusohjelma ”OpenHaptics toolkit” kansiosta. Asennusohjelma asentaa GLUT kirjaston oletuksena, kirjastoa tarvitaan graafisissa esimerkeissä. Asennusohjelmassa on kuitenkin mahdollisuus olla asentamatta kirjastoja. Asennuksen jälkeen laajempi dokumentaatio löytyy asennuksen juuri kansion alta kansiosta ”doc” (PROGRAMMER'S GUIDE 2004).

Kehitysympäristönä on Microsoft Visual C++ .NET 2003. Tämän takia joudutaan Haptic Device Utility (HDU) ja Haptic Library Utility (HLU) kirjastot kääntämään uudestaan, koska kirjastot ovat alunperin käännetty Visual C++ 6.0:lle.

Kirjastojen kääntäminen suoritetaan seuraavasti:

1. Mennään OpenHaptics utilities kansioon (oletuksena C:\ProgramFiles\OpenHaptics\utilities).
2. Mennään kansioon utilities\src\HDU ja aukaistaan HDU.dsw Visual C++ .NET 2003:lla. Ohjelma kysyy haluatko vaihtaa projektin uudempaan *.sln muotoon, tähän vastataan kyllä.
3. Käännetään Release ja Debug versiot HDU kirjastosta. Tämä onnistuu käyttämällä Build/Batch Build valikosta ja painamalla sieltä Build nappia.
4. Kopioidaan uudelleen käännetyt hdu.lib ja hdud.lib tiedostot Release ja Debug kansioista src/HDU kansioon utilities/lib. Tällöin kirjoitetaan tiedostojen päälle, jotka ovat käännetty Visual C++ 6.0:lle. (hdud.lib on debug ja hdu.lib on release).
5. Samat toiminnot toistetaan HLU, HapticMouse ja SnapConstraints kirjastoille (README 2005).

2.3 Kehitysympäristön asetukset

Käytettävät kirjastot kopioidaan kansioon, josta kehitysympäristö hakee oletuksena käytettäviä kirjastoja. Alla on taulukko jossa näkyy mitä pitää kopioida ja mihin kansioon.

Koska kirjastot kopioidaan oletus kansioon, ei kehitysympäristöön tarvitse asettaa polkuja eri kirjastoille.

Taulukko 4: Kirjatojen kopiointi, haptiikkalaite ((PROGRAMMER'S GUIDE 2004).

Mistä	Mitä	Minne
C:\Program Files\SensAble \3DTouch\include\	Kansion sisältö	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include\
C:\Program Files\SensAble \3DTouch\utilities\include\	Kansion sisältö	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include\
C:\Program Files\SensAble\3DTouch\lib\	hd.lib ja hl.lib	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\lib\
C:\Program Files\SensAble\3DTouch\utilities\lib\	hdu.lib, hdud.lib, hlu.lib ja hlud.lib	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\lib\
C:\Program Files\SensAble\3DTouch\lib\	hd.dll ja hl.dll	C:\WINDOWS\system32\
C:\Program Files\SensAble\3DTouch\utilities\lib\	glut32.dll	C:\WINDOWS\system32\

Seuraavaksi asennetaan GLFW toimimaan Microsoft Visual Studio .NET 2003 –kehitysympäristössä. GLFW on alustariippumaton ohjelmointi rajapinta, jolla hallitaan käyttöjärjestelmien eri tehtäviä. Näitä ovat esimerkiksi:

- ikkunoinnin hallinta ohjelmoidessa OpenGL™:llä
- käyttäjän syötteet näppäimistö, hiiri ja ilotikku
- ajastin
- tuki moniajoon
- tuki tekstuurien latautselle (Reference Manual 2007).

Lähdekoodi GLFW-kirjastolle saadaan projektin kotisivulta http://glfw.sourceforge.net/. Ladatun paketti sisältää ohjeen kirjaston kääntämiseen, joka on seuraavanlainen.

1. Avataan komentorivi ja mennään kansioon jonne GLFW:n lähdekoodi on purettu.
2. Suoritetaan tiedosto nimeltään VCVARS32.BAT komentorivillä. Kirjoitetaan

komentoriville ”C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\bin\ VCVARS32” (heittomerkit sisältyvät komentoon).

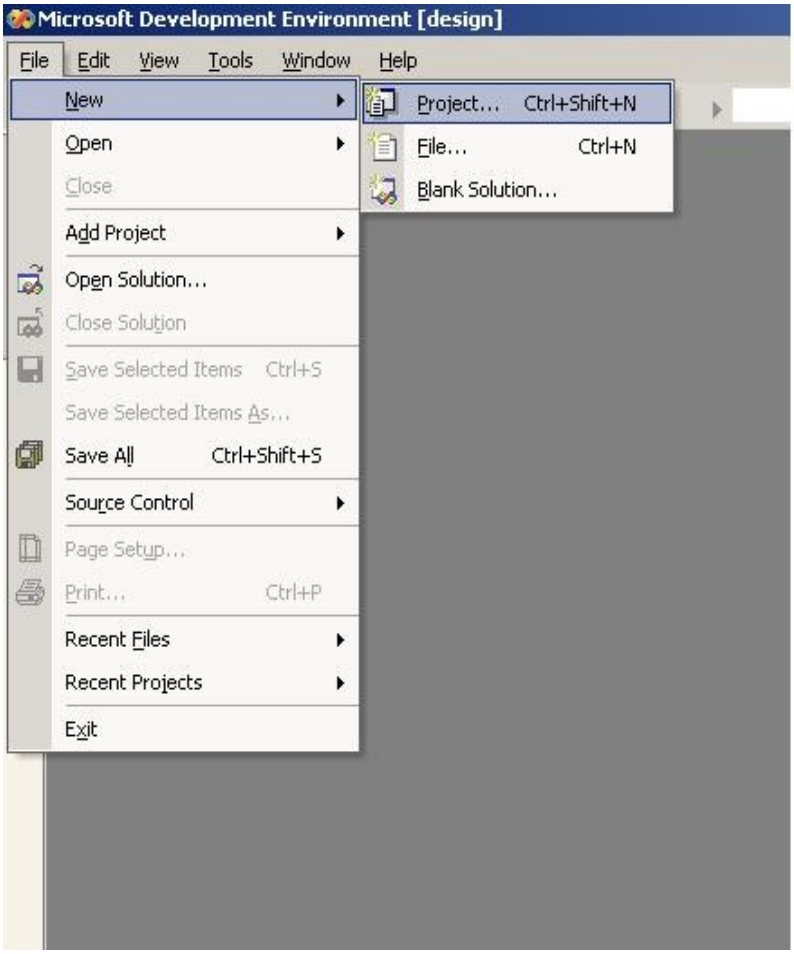
3. Kirjoita nmake win32.msvc (GLFW 2007).

Kääntämisen jälkeen kolme kirjastoa ilmetyy lib\win32 kansioon. glfw.lib, glfwdll.lib ja glfw.dll, nämä kopioidaan alla olevan taulukon mukaisesti (GLFW, 2007).

Taulukko 5: Kirjastojen kopiointi GLFW (GLFW, 2007).

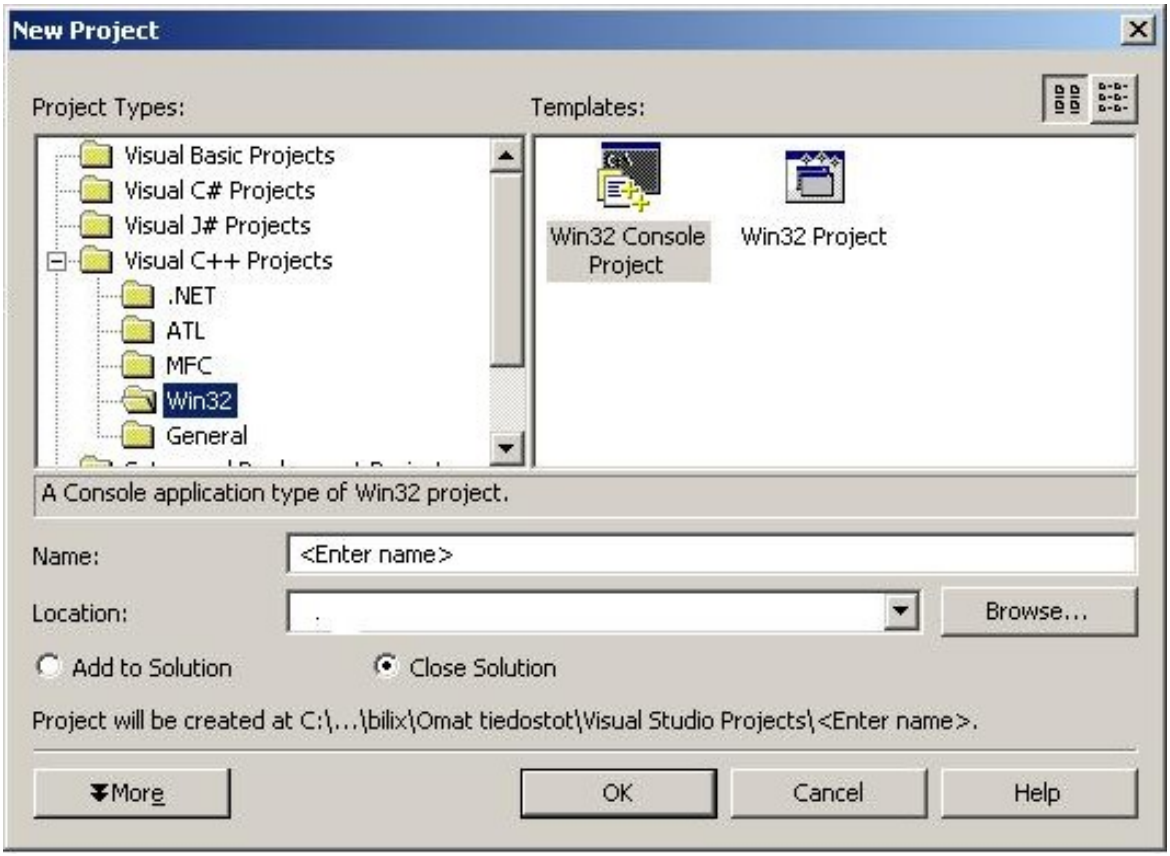
Mikä	Minne
glfw.lib ja glfwdll.lib	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\lib\
glfw.dll	C:\WINDOWS\system32\
glfw.h	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include\GL\

Seuraavaksi tehdään uusi projekti kehitysympäristöön. Tämä tehdään alla kuvan 6 osoittamalla tavalla.



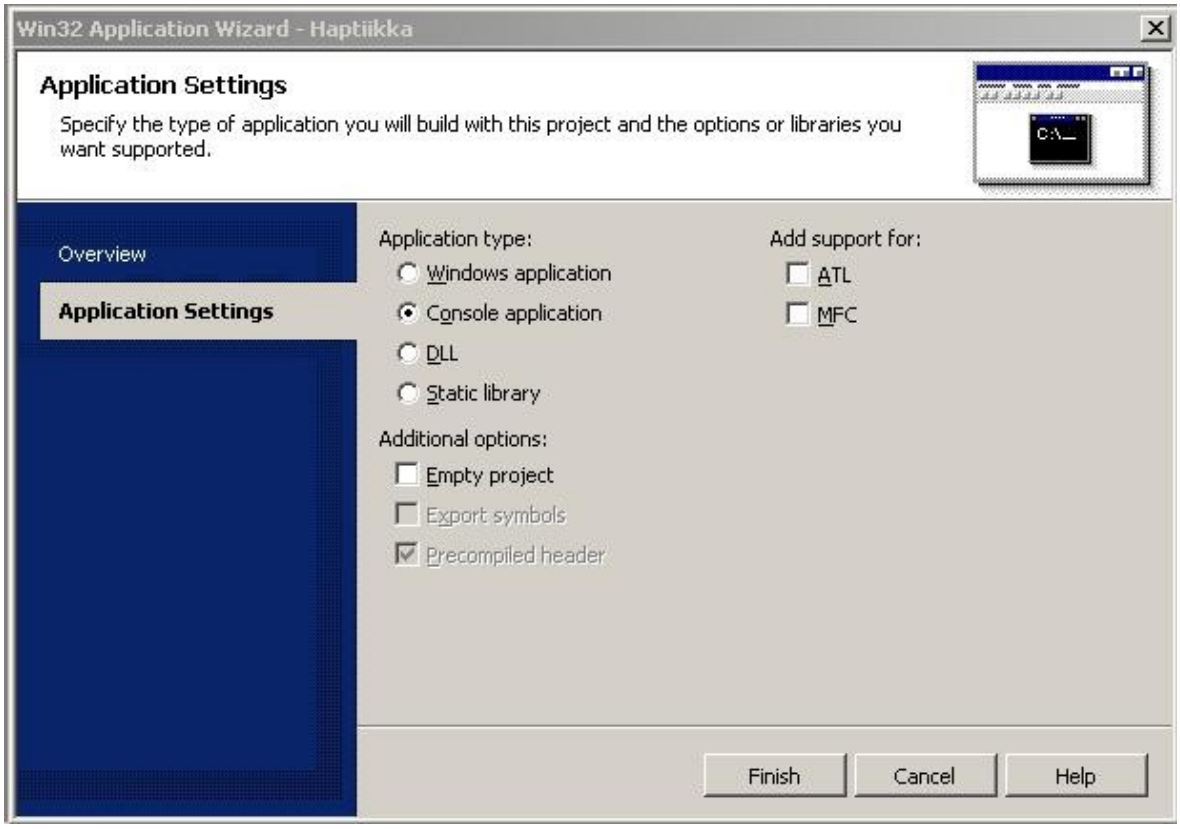
Kuva 6: Uuden projektin aloittaminen.

Tehtävä ohjelma ei sisällä grafiikkaa, joten valitaan projektiksi ”Win32 Console Project”, kuva 7 alla.



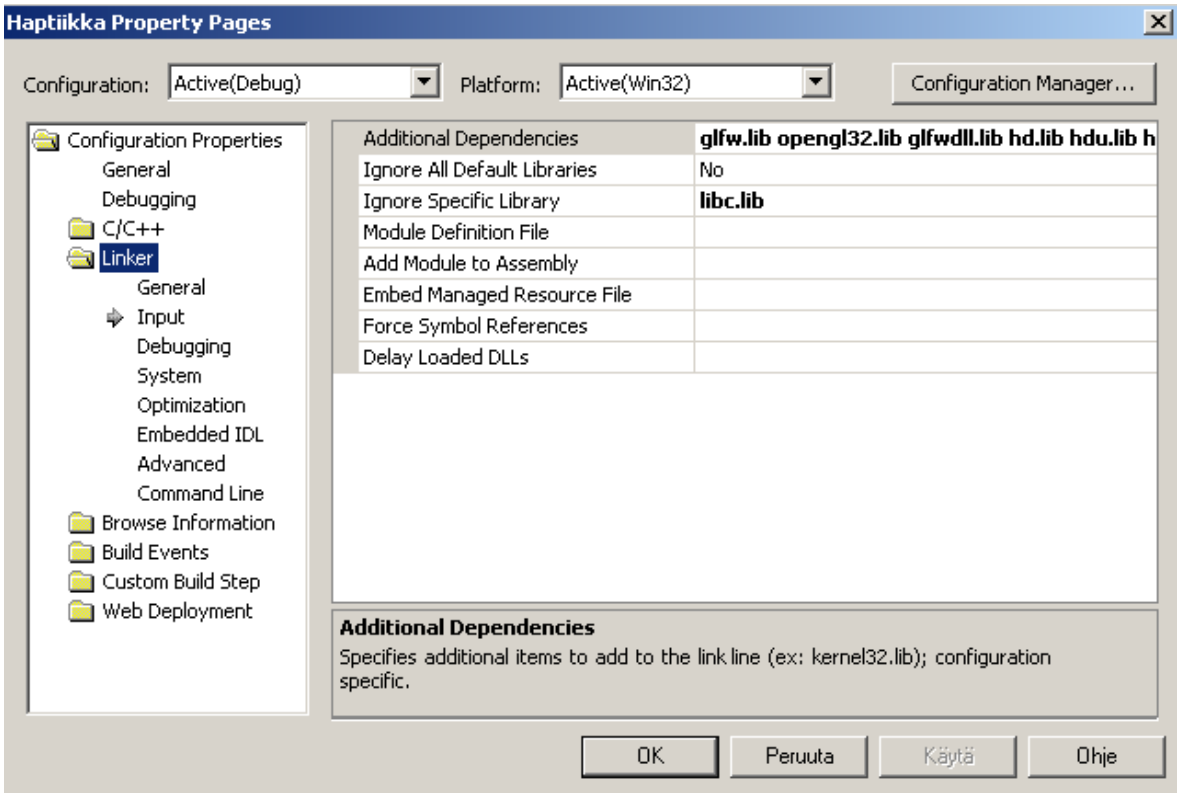
Kuva 7: Valitaan projektiksi "Win32 Console Project".

Tämän jälkeen täytyy vielä määrittää projektin asetukset, jotka ovat esitetty alla kuvassa 8.



Kuva 8: Projektin asetukset.

Seuraavaksi asetetaan linkitettävät kirjastot projektin asetuksiin. Alla kuvassa 9 on esitetty minne lisätään kirjastot joita tarvitaan lähdekoodin kääntämiseen.



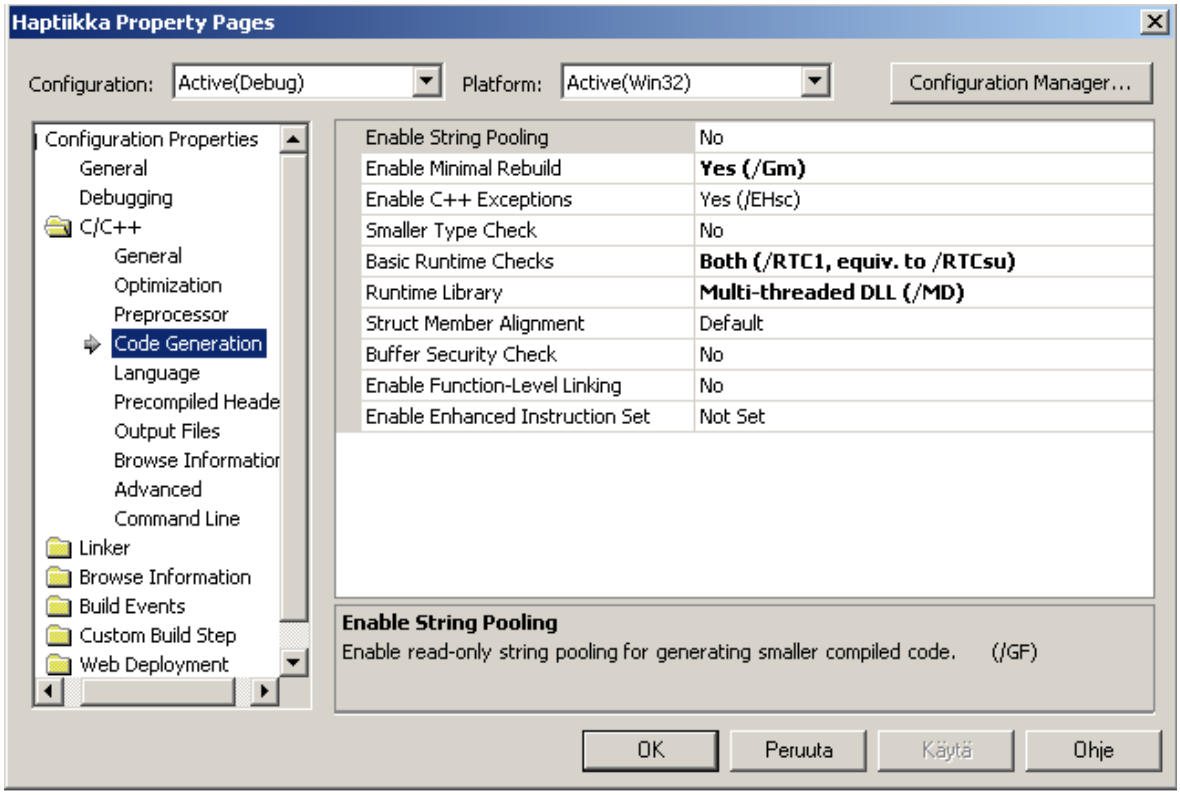
Kuva 9: Välilehti, jonne laitetaan linkitettävät kirjastot.

Alla on lista kirjastoista, jotka laitetaan kuvassa 9 esitettyyn välilehden ylimpään kohtaan ”Additional dependencies”:

- glfw.lib
- opengl32.lib
- glfwdll.lib
- hd.lib
- hdu.lib
- hdud.lib (PROGRAMMER'S GUIDE 2004).

Yhtä kirjastoa ei tarvita kääntämisessä, koska se aiheuttaa seuraavan varoituksen ”Haptiikka warning LNK4098: defaultlib 'LIBC' conflicts with use of other libs; use /NODEFAULTLIB:library”. Eli oletksena oleva LIBC-kirjasto on ristiriidassa joidenkin toisten kirjastojen kanssa. Virheilmoitusta ei tule kun kirjastoa libc.lib ei käytetä käännettäessä. Kyseistä kirjastoa ei käytetä, kun se kirjoitetaan kohtaan ”Ignore Specific Library”.

Seuraavaksi täytyy muuttaa välilehdelle, joka on esitetty alla kuvassa 10, kohtaan ”Runtime Lirary” ”Multi-threded DLL (/MD)” (PROGRAMMER'S GUIDE 2004).



Kuva 10: Tähän keksi joku fiksu juttu.

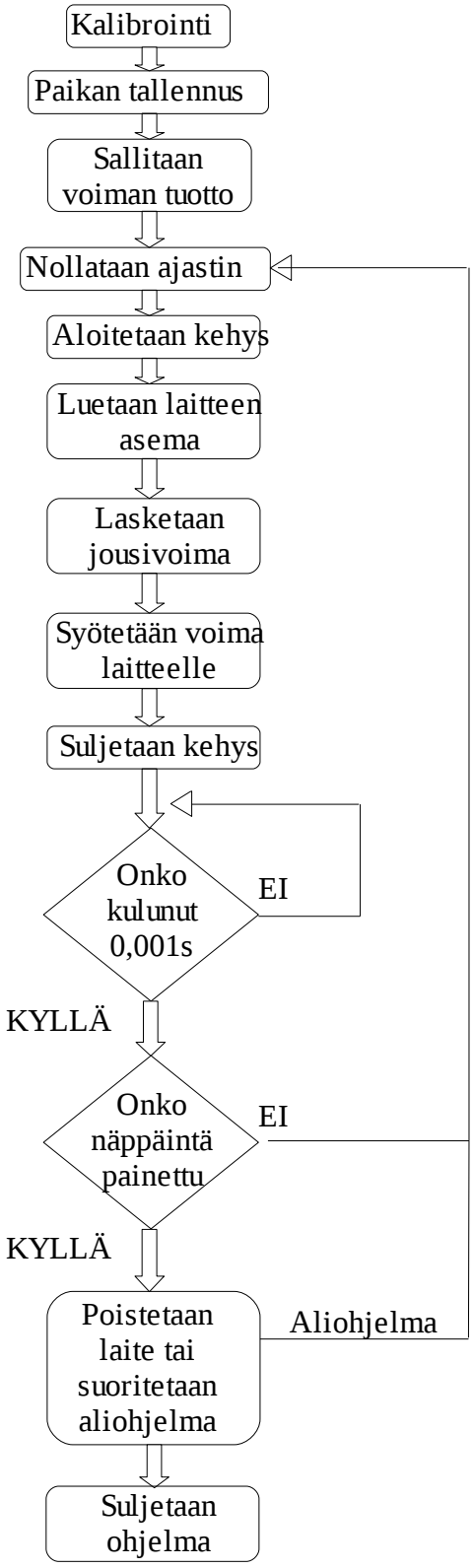
Nyt kehitysympäristön asetukset ovat valmiit sovelluskehitystä varten.

2.4 Sovelluskehitys

Ohjelmointi tapahtuu matalantason ohjelmointina. Rajapintana toimii OpenHaptics toolkit:n mukana tullut Haptic Device API (HDAPI). Tämä antaa matalantason käskyt haptiikka laiteelle, jolloin voima täytyy laskea itse ja tämän jälkeen se lähetetään laitteelle. HDAPI:sta löytyy myös aikataulutin, jolla voidaan määrittää tehtävien prioriteetit ja suoritustaajuus.

Ohjelmassa käytetään laitteen mukana tulleita kirjastoja laitteen hallintaan ja GLFW-kirjastolla saadaan ajastettua tehtäviä ja luettua näppäimistöä. Laitteen voiman ja paikan päivitys täytyy tehdä vähintään 1000Hz taajuudella, ettei laitteen tuottama voima ole

tärisevä. Alla kuvassa 11 on esitetty käytettävä ohjelmarakenne.



Kuva 11: Ohjelman rakenne.

Kalibrointi tehdään aina, kun laite on ollut pois kytkettynä verkkovirrasta. Kalibrointi aloitetaan alustamalla laite

hdInitDevice(HD_DEFAULT_DEVICE) (1),

jolloin laite tallentuu muuttujaan *HD_DEFAULT_DEVICE*. Kun laitetta käsitellään ohjelmassa alustuksen jälkeen käytetään edellä esitettyä muuttujaa. Kun tämä on tehty voidaan laiteelta hakea tuettu kalibrointitapa funktiolla

hdGetIntegerv(HD_CALIBRATION_STYLE, &TuetutKalibrointiTavat) (2).

Tämä täytyy tehdä, jos ei tiedä laiteen tukemaa kalibrointi tapaa, koska jotkut laitteista tukevat useampaa kalibrointitapaa. Hankittu laite tukee kuitenkin vain manuaalista kalibrointitapaa. Tässä tapauksessa kalibrointi tehdään asettamalla laitteen varret kohtisuoraan toisiaan vastaan, jolloin laite on nollapisteessään. Kalibrointitapa voidaan varmistaa vertaamalla. onko tuettu kalibrointityyli *HD_CALIBRATION_ENCODER_RESET*

if (TuetutKalibrointiTavat == HD_CALIBRATION_ENCODER_RESET) (3).

Ohjelmassa kalibrointi tehdään funktiolla

hdUpdateCalibration(TuetutKalibrointiTavat,) (4)

ja kalibroinnin tarkastus suoritetaan funktiolla *hdCheckCalibration()*

if (hdCheckCalibration() == HD_CALIBRATION_OK) (5),

joka palauttaa *HD_CALIBRATION_OK*, jos kalibrointi on onnistunut.

2.4.1 Jousivoiman laskenta

Aina, kun käsitellään laitetta eli haetaan paikkatieto tai asetetaan voima, täytyy se tehdä kehyksen sisällä. Kehyksen alkaessa otetaan laitteen tila muistiin ja laskiessa kehyksen sisällä arvot pysyvät vakioina. Kehys aloitetaan seuraavalla funktiolla

hdBeginFrame(hdGetCurrentDevice()) (6)

Paikkatieto saadaan funktiolla

hdGetDoublev(HD_CURRENT_POSITION, paikka) (7),

jossa paikkatieto tallennetaan muuttujaan *paikka*. Alussa alustetaan muuttuja *ankkuri* nollaksi, koska laskenta aloitetaan heti kalibroinnin jälkeen, jolloin laite on vielä nollapisteessään. Kun paikkatieto ollaan saatu voidaan laskea etäisyys *x* ankkuri pisteestä. Tämän jälkeen saadaan laskettua jousivoima *F* kaavalla

$$F = k \cdot x \quad (8),$$

jossa *k* on jousivakio. Voiman laskenta ohjelmassa suoritetaan kahdella funktiolla. Ensiksi laskentaan nykyisen paikan etäisyys pisteestä ankkuri.

$$x = ankkuri - paikka \quad (9).$$

Vähennyslaskussa otetaan huomioon voiman etumerkki tässä vaiheessa. Laskenta ohjelmassa tapahtuu funktiolla *hduVecSubtract()*, jossa muuttujien *ankkuri* ja *paikka* erotus sijoitetaan muuttujaan *voima*.

hduVecSubtract(voima, ankkuri, paikka) (10).

Seuraavaksi laskentaan *voima*, jonka laitteen siirtäminen kohdasta *ankkuri* tuottaa. Tämä tapahtuu seuraavalla funktiolla, jossa muuttujan *voima* ja *Jousivoima*, tulos sijoitetaan muuttujaan *voima*.

hduVecScaleInPlace(voima, Jousivoima) (11).

Itse voiman asettaminen laitteelle tapahtuu seuraavalla funktiolla

hdSetDoublev(HD_CURRENT_FORCE, voima) (12).

2.4.2 Voiman päivittäminen laitteelle käyttäen GLFW-kirjastoa

Voiman suunta ja suuruus täytyy päivittää laitteelle vähintään 1000Hz taajuudella, ettei voimatakaisinkytkentä ole tärisevä. Oikealla taajuudella toimiva silmukka saadaan aikaan GLFW:n tarjoamalla laskurilla. Laskuria nollaamalla ja odottamalla haluttu aika, saadaan aikaan laitteen päivitys halutulla taajuudella. Laskuri lähtee käyntiin samalla, kun alustetaan itse GLFW. Laskuri täytyy täytyy nollata jokaisella laitteen päivitys kerralla. Tämä rakenne toteutetaan kuvassa 6 esitetyllä tavalla.

Laite ei kuitenkaan toiminut halutulla tavalla. Asia käsitellään tarkemmin kohdassa 3 Tulokset ja 4 Tulosten tarkastelu. Ohjelman koodi löytyy liitteestä 1.

2.4.3 Voiman päivittämien käytten OpenHaptics toolkit:n aikataulutinta

Toisen päivityssilmukan tekemiseen käytin OpenHaptics toolkit:n tarjoamaa aikataulutinta. Voiman laskenta suoritetaan samalla tavalla kuin GLFW:llä tehdyssä päivityssilmukassa, mutta laite päivitetään käyttäen asynkronista tehtävää. Tämä tehtävä suoritetaan jokaisella aikatauluttajan tuottamalla kierroksella, jossa taajuus on oletuksena 1000 Hz.

Toinen mahdollinen kutsu aikatauluttajalle on synkroninen kutsu. Synkronisen kutsun avulla voidaan tehdä laitteen lukeminen ja arvojen muuttaminen synkronisesti muiden tehtävien kanssa. Näin vältetään jaetundatan ongelma. Jaetundatan ongelma syntyy silloin, kun toinen tehtävä pääsee muuttamaan dataa, toisen tehtävän käyttäessä sitä, samasta

muistiosoitteesta.

2.4.3.1 Asynkroninen tehtävä

Tehtävän esitteleminen aikatauluttajalle suoritetaan seuraavasti

HDSchedulerHandle hdScheduleAsynchronous (HDSchedulerCallback
*pTehtava,void *pKayttajantieto*
Hdushort Prioriteetti) (14).

Tehtävä palauttaa *HDSchedulerHandle* tyyppisen muuttujan, jota voidaan käyttää tarkistamaan onko tehtävä vielä aktiivisena eli suoritetaanko tehtävä seuraavalla kierroksella tai odottamaan, että asynkroninen tehtävä suoritetaan loppuun. Parametrit, jotka annetaan esittelyvaiheessa ovat seuraavat:

- *Tehtävä* aliohjelma, joka suoritetaan kutsussa
- *pKayttajantieto* on tieto jota voidaan käyttää aliohjelmassa
- *Prioriteetti* tehtävän prioriteetti, tehtävät suoritetaan prioriteetin mukaisessa järjestyksessä.

Esittelyn jälkeen voidaan tarkastaa onko tapahtunut virhettä. Virheet, jotka voivat tapahtua ovat esitelty alla taulukossa 6.

Taulukko 6: Asynkronisen tehtävän esittelyssä mahdollisesti saatavat virheilmoitukset (PROGRAMMER'S GUIDE 2004).

Palautusarvo	Virhe
HD_SCHEDULER_FULL	Aikatauluttajan tukemien tehtävien lukumäärä täynnä
HD_INVALID_PRIORITY	Annettu liian suuri prioriteetin arvo. Täytyy olla välillä HD_MIN_SCHEDULER_PRIORITY - HD_MAX_ SCHEDULER_PRIORITY

2.4.3.2 Synkroninen tehtävä

Synkroninen tehtävä esitellään seuraavalla tavalla

void

hdScheduleSynchronous(HDSchedulerCallback pTehtava,

*void *pKayttajantieto*

Hdushort Prioriteetti) (15).

Esittely on muuten sama kuin asynkronisessa, mutta tyyppi on *void*. Annettavat parametrit ovat vastaavat kuin asynkronisessa kutsussa. Myös virheentarkastuksessa on samat virheet kuin asynkronisessa, jotka ovat taulukossa 6.

Synkroninen tehtävä ei käynnisty automaattisesti niin kuin asynkroninen vaan synkronista tehtävää kutsutaan aina tarvittaessa. Synkronisella tehtävällä voidaan käsitellä esimerkiksi näppäimistön lukua (PROGRAMMER'S GUIDE 2004).

Aikataulutin käynnistetään funktiolla

hdStartScheduler() (16).

Virhe joka voi tulla käynnistettäessä aikataulutinta on *HD_TIMER_ERROR*, joka syntyy jos aikataulutinta ei käynnisty tai aikatauluttimella olevaa tehtävää ei voitu alustaa. Aikataluttajan lopettaminen tapahtuu funktiolla

hdStopScheduler() (17).

Ohjelmassa on yksi asynkroninen tehtävä jolla päivitetään laitetta. Tehtävässä lasketaan voima ja lähetetään se laitteelle ja tarkastetaan onko tapahtunut virhe suoritettaessa tehtävää. Toinen tehtävä, jota käytetään on synkroninen. Tämän tehtävän avulla muutetaan käytettävää jousivakiota. Näin vältetään jaetundatan ongelma.

Synkroniseen tehtävään annetaan parametrina osoitin muuttujaan, joka sijoitetaan muuttujaan *Jousivakio*, jonka jälkeen palautetaan tieto, että tehtävä suoritettu. Synkronisen tehtävät kutsussa annetaan tehtävän prioriteetiksi *HD_DEFAULT_SCHEDULER_PRIORITY*, joka on jokin arvo minimin ja maksimin välillä. Tällöin ei välitetä milloin tehtävä tehdään kunhan se tehdään kyseisellä aikatauluttajan kierroksella (PROGRAMMER'S GUIDE 2004).

Asynkronista tehtävää suoritetaan koko ajan taustalla ja synkroninen suoritetaan aina, kun painetaan nuolinäppäintä ylös tai alas. Tämän ohjelman koodi löytyy liitteestä 2.

3 Tulokset

Hankittu laite täyttää sille asetetut vaatimukset voiman, vapausasteiden ja työtilan suhteen. Myös laitteen ohjelmisto toimii sille vaaditussa käyttöjärjestelmässä.

Laiteella saadaan tuotettua todellisen tuntuinen tuntopalaute, mikä saatiin tuotettua edellä tehdyllä ohjelmalla.

Laite ei kuitenkaan toiminut niin kuin olisi pitänyt, kun käytettiin GLFW:n tarjoamaa ajastinta. Laite ei päivittynyt jokaisella silmukan kierroksella. Välillä laitteen pää pääsi liikkumaan huomattavasti pois nollapisteestä, jolloin voima tuli nykivänä. Tämä ilmiö ei kuitenkaan toistunut säännöllisesti vaan se tuli satunnaisesti.

Käyttämällä Open Haptic toolkitin tarjoamaa aikataulutinta päivittämään laitteen tuottamaa voimatakaisinkytkentää, oli voima realistisen tuntuinen.

4 Tulosten tarkastelu

Testatakseni johtuiko satunnainen nykiminen päivitys taajuudesta tulostin juoksevan ajan komentoriville jokaisella kierroksella. Ajan takeltelemisen olisi huomannut, koska laitteen asemaa kerkesi siirtämään pois nollapisteestä noin 50 mm etäisyydelle. Aika ei kuitenkaan pysähtynyt, mutta laitteelle ei kuitenkaan päivittynyt voima. Luultavasti tiedonsiirto ei toiminut aina laitteen ja tietokoneen väillä.

5 Johtopäätökset

Hankittu laite tuottaa realistisen voimatakaisinkytkennän, mutta se ei onnistunut käyttämällä GLFW:n tarjoamaa ajastinta. GLFW:n avulla tehtyyn päivityssilmukkaan olisi voinut lisätä virheen tarkastuksen, jolloin olisi voinut saada selville minkä takia laitteen päivitys ei onnistunut. Laitteelta olisi voinut tulla joku seuraavista virheistä:

- *HD_FORCE_ERROR*, voiman lähetys epäonnistui
- *HD_COMM_ERROR*, virhe yhteydessä
- *HD_DEVICE_FAULT*, laite ei ole kytkettynä tai päällä (PROGRAMMER'S GUIDE 2004).

Kuitenkin laitteen hallinta on helpompaa käyttäen OpenHapticToolkit:n tarjoamaa

aikataulutinta, koska sillä saadaan asetettua prioriteetit eri tehtäville ja saavutetaan laitteen turvallinen hallinta.

LÄHTEET

GLFW. Readme.html. Updated 5.6.2007. [viitattu 7.1.2008]
Saatavissa http://sourceforge.net/project/downloading.php?
groupname=glfw&filename=glfw-2.6.zip&use_mirror=surfnet

Features [GLFW-projektin www-sivuilla] [verkkodokumentti]. [viitattu 2.1.2008]
Saatavissa http://glfw.sourceforge.net/

Inition. Updated 19.4.2005. [www-tuotedokumentti] [viitattu 29.1.2008].
Saatavissa http:/http://www.inition.com/inition/product.php?
URL_=product_ffhaptic_forcedimension_3dofomega&SubCatID_=36
rinnakkaisrakenne

Inition. Updated 13.11.2006. [www-tuotedokumentti] [viitattu 29.1.2008].
Saatavissa http://www.inition.com/inition/product.php?
URL_=product_ffhaptic_sensable_phantomdesktop&SubCatID_=36
käsivarsi

Inition. 2008a [www-tuotedokumentti] [viitattu 29.1.2008].
Saatavissa http://www.inition.co.uk/inition/product.php?
URL_=product_glove_vti_touch&SubCatID_=26

Inition. 2008b [www-tuotedokumentti] [viitattu 29.1.2008].
Saatavissa http://www.inition.com/inition/product.php?
URL_=product_ffhaptic_immersion_cyberforce&SubCatID_=37
häkkyrä

Sensable. PHANTOM™ PREMIUM User's Guide. Updated 18 November 2004, [viitattu 20.10.2007].
Saatavissa SensAble Developer Support Center, vaatii salasanan.

README.txt. Updated 28 July 2005 [viitattu 22.10.2007].

Saatavissa SensAble Developer Support Center, vaatii salasanan.

Sensable. OPENHAPTICS™ TOOLKIT VERSION 1.0 PROGRAMMER'S GUIDE. Updated 15 August 2004, [viitattu 22.11.2007].
Saatavissa SensAble Developer Support Center, vaatii salasanan.

Lava Computer MFG Inc. IEEE 1284: Parallel Ports. [www-tuotedokumentti]. [viitattu 8.1.2008].
Saatavissa http://www.nor-tech.com/solutions/dox/ieee1284_parallel_ports.pdf

Reference Manual. [GLFW:n www-sivuilla]. Updated September 1, 2007. [viitattu 11.1.2008]
Saatavissa http://glfw.sourceforge.net/

LIITE 1. Lähdekoodi

```
#include "stdafx.h"
#include <conio.h>

//nämä ovat laitteelle
#include <HD/hd.h>                          //pääkirjasto, sisältää kaikki HD-kansion kirjastot
#include <HDU/hduError.h>
#include <HDU/hduVector.h>
#include <GL/glfw.h>



int _tmain(int argc, _TCHAR* argv[])
{
  int TuetutKalibrointiTavat, KalibrointiTapa, merkki;
  double  aika=0.0;
  HDdouble MaxJaykkyys = 1, muutos = 0.01;
  HDErrorInfo error;
  HDdouble Jousivakio = 0.1;
  HDdouble Jousi = Jousivakio;
  HHD hHD;

    hduVector3Dd ankkuri;
    hduVector3Dd paikka;
    hduVector3Dd voima;
          voima[0]=0;
          voima[1]=0;
          voima[2]=0;


          ankkuri[0]=0;
          ankkuri[1]=0;
          ankkuri[2]=0;

  //KALIBROINTI tarvitsee suorittaa vain silloin, kun laite on ollut pois kytkettynä verkkovirrasta.
  /*
     Laitteen alustus täytyy tehdä ennen kuin kutsutaan jotakin hd funktiota.
     Alustuksessa on oletuksena kyktketty voimatakaisin kytkentä pois.
     Voimatakaisinkytkentä laitetaan päälle funktiolla:
     hdEnable(HD_FORCE_OUTPUT)
  */

          // GLFW:n alustus
  glfwInit();

          //alustetaan kaytettava haptiikka laite
  hHD = hdInitDevice(HD_DEFAULT_DEVICE);


if (HD_DEVICE_ERROR(error = hdGetError()))
    {
      hduPrintError(stderr, &error, "Laitteen alustus epäonnistui\n");
      fprintf(stderr, "\nPaina mitä tahansa näppäintä lopettaaksesi.\n");
```

```
    getch();
    return -1;

  }
 //Tervehdys teksti käyttäjälle, jossa haetaan laitteen tiedot ja tulostetaan se.

 printf("Tere haptikoija!\n");
 printf("Löydettiin laite: %s.\n\n", hdGetString(HD_DEVICE_MODEL_TYPE));
 printf("\n\n\nSeuraavaksi suoritetaan kalibrointi.\n");



 //Haetaan tuettu kalibrointi tapa.
 hdGetIntegerv(HD_CALIBRATION_STYLE, &TuetutKalibrointiTavat);

 if (TuetutKalibrointiTavat & HD_CALIBRATION_ENCODER_RESET)
  {
   KalibrointiTapa = HD_CALIBRATION_ENCODER_RESET; //tämä se varmaan on premium
1.5:llä
  }
 else
 {
        printf("Virhe kalibroitaessa");
        hdDisableDevice(hHD);
        glfwTerminate();
        getch();
        return 0;

 }


if (KalibrointiTapa == HD_CALIBRATION_ENCODER_RESET)
  {
    printf("Suoritetaan manuaalinen kalibrointi.\n");
    printf("Aseta laitteen varret kohtisuoraan toisiaan vastaan\n\n");
    printf("Paina nappia,kun olet asettanut laitteen oikeaan asentaoon\n");

    getch();

    hdUpdateCalibration(KalibrointiTapa);
        }
        if(HD_CALIBRATION_OK == hdCheckCalibration())  printf("Kalibrointi OK\n");

        else
        {
                //jos kalibtointi ei onnistunut poistutaan ohjelmasta
                printf("Kaslibrointi ei onnistunut. Paina jotain nappainta\n");
                //poistetaan laite
                hdDisableDevice(hHD);
                //suljetaan GLFW
                glfwTerminate();
                getch();
                return 0;
```

```
        }


  //maksimi jäykkyys jota laite voi käsitellä, ettei ylitetä jäykkyyttä, koska jousivakiota voi muutta
  hdGetDoublev(HD_NOMINAL_MAX_STIFFNESS, &MaxJaykkyys);

  printf("\nPaina näppäintä lopettaaksesi ohjelma.\n");



  //sallitaan voiman tuotto
  hdEnable(HD_FORCE_OUTPUT);


while(1)
   {
    glfwSetTime(0);
     //OHJE


          //kehyksen alku
          hdBeginFrame(hdGetCurrentDevice());
          hdGetDoublev(HD_CURRENT_POSITION, paikka);


    //vähennetään ankkurista paikka ja sijoitetaan muuttujaan voima
    hduVecSubtract(voima, ankkuri, paikka);

    //Kerrotaan voima Jousivakiolla ja sijoitetaan se muuttujaan voima
    hduVecScaleInPlace(voima, Jousivakio);

    //Asetetaan voima laitteelle.
    hdSetDoublev(HD_CURRENT_FORCE, voima);

    //kehyksen loppu
    hdEndFrame(hdGetCurrentDevice());
        //lisataan aikaan millisekunti
        aika += 0.001;
        //tulostetaan aika silmukan sisalla, etta nahdaan kierretaanko silmukkaa reaaliajassa.
        printf("%.3f \n", aika);


    //Jos nappia painettu poistutaan tai lisätään/vähennetään jousen juosivakiota.

          if(_kbhit())
           {
                  break;
                merkki = getch();

                if(('+' == merkki))
                        {
                        Jousivakio = Jousivakio + muutos;
```

```
                    //ei anneta jousivakion kasvaa suuremmaksi kuin laitteen maksimi jaykkyys
                    if (Jousivakio > MaxJaykkyys)
                        Jousivakio = MaxJaykkyys;

                    }
                else if(('-' == merkki))
                    {
                    Jousivakio = Jousivakio - muutos;
                    //ei lasketa jousivakiota negatiiviseksi, ettei voiman suunta muutu
                    if(Jousivakio < 0.01)
                        Jousivakio = 0.01;
                    }
                else break;

            }
        //silmukka, jos ei ole kulunut yli 0,001 sekuntia
        while(glfwGetTime()<=0.001);
        }


    //Kun poistutaan silmukasta, pitää kytkeä pois haptinen laite ja sammuttaa GLFW
    hdDisableDevice(hHD);
    glfwTerminate();


return 0;
}
```

LIITE 2. Lähdekoodi

```
#include "stdafx.h"
#include <conio.h>


//Laitteen vaatimat kirjastot
#include <HD/hd.h>                      //Paakirjasto, sisaltaa kaikki HD-kansion kirjastot
#include <HDU/hduError.h>          //Virheet
#include <HDU/hduVector.h>              //Vektoreiden lakentaan

//Nappaimiston lukua varten
#include <GL/glfw.h>

//Globaalit muuttujat, ei tule jaetundatan ongelmaa, koska kaytetaan aikatauluttajaa
static HDdouble Jousivakio = 0.1;
HDSchedulerHandle gCallbackHandle = 0;

//aikatauluttajan kutsumat tehtavat
HDCallbackCode HDCALLBACK JousivoimanAsettaminen(void *p_Kayttajan_tieto);
HDCallbackCode HDCALLBACK JousivoimanLaskeminen(void *p_Kayttajan_tieto);


int _tmain(int argc, _TCHAR* argv[])
{
  int TuetutKalibrointiTavat, KalibrointiTapa;
  double  aika=0.0;
  double muutos = 0.01;
  HDErrorInfo error;
  HDdouble Jousi = Jousivakio;
  HDdouble MaxJaykkyys = 1;

  //muuttuja johon otetaan kaytettava laite talteen
  HHD hHD;


  //KALIBROINTI tarvitsee suorittaa vain silloin, kun laite on ollut pois kytkettyna verkkovirrasta.
  /*
     Laitteen alustus taytyy tehda ennen kuin kutsutaan jotakin hd funktiota.
     Alustuksessa on oletuksena kyktketty voimatakaisin kytkenta pois.
     Voimatakaisinkytkenta laitetaan paalle funktiolla:
     hdEnable(HD_FORCE_OUTPUT)
  */

//GLFW alustus
  glfwInit();


//Taytyy aukaista ikkuna jotta nappaimiston lukeminen onnistuu
//aukaistaan mandollisimman pieni ikkuna
  glfwOpenWindow( 1, 1, 8, 8, 8, 8, 16, 0, GLFW_WINDOW );

  hHD = hdInitDevice(HD_DEFAULT_DEVICE);
```

```
if (HD_DEVICE_ERROR(error = hdGetError()))
    {
     hduPrintError(stderr, &error, "Laitteen alustus epaonnistui\n");
     fprintf(stderr, "\nPaina mita tahansa nappainta lopettaaksesi.\n");
     getch();
     return -1;

    }
  //Tervehdys teksti kayttajalle, jossa haetaan laitteen tiedot ja tulostetaan se.

  printf("Tere haptikoija!\n");
  printf("Loydettiin laite: %s.\n\n", hdGetString(HD_DEVICE_MODEL_TYPE));
  printf("\n\n\nSeuraavaksi suoritetaan kalibrointi.\n");




  //Haetaan tuettu kalibrointi tapa.
  hdGetIntegerv(HD_CALIBRATION_STYLE, &TuetutKalibrointiTavat);

  if (TuetutKalibrointiTavat & HD_CALIBRATION_ENCODER_RESET)
    {
     KalibrointiTapa = HD_CALIBRATION_ENCODER_RESET;
    }
  else
  {
            printf("Valitse eri kalibrinti tapa");
            hdDisableDevice(hHD);
            glfwTerminate();
            getch();
            return 0;

  }

if (KalibrointiTapa == HD_CALIBRATION_ENCODER_RESET)
    {
     printf("Suoritetaan manuaalinen kalibrointi.\n");
     printf("Aseta laitteen varret kohtisuoraan toisiaan vastaan\n\n");
     printf("Paina ENTER ,kun olet asettanut laitteen oikeaan asentaoon\n");

     while(!glfwGetKey(GLFW_KEY_ENTER) == GLFW_PRESS) glfwPollEvents();

     hdUpdateCalibration(KalibrointiTapa);
          }
          if(HD_CALIBRATION_OK == hdCheckCalibration())  printf("Kalibrointi OK\n");

          else
          {
                    printf("Kalibrionti epaonnistui\n");
                    hdDisableDevice(hHD);
                    glfwTerminate();
                    getch();
                    return 0;
```

```
        }



  //maksimi jaykkyys jota laite voi kasitella, ettei yliteta jaykkyytta, koska jousivakiota voi muutta
  hdGetDoublev(HD_NOMINAL_MAX_STIFFNESS, &MaxJaykkyys);

  printf("\nPaina nappainta Q lopettaaksesi ohjelman.\n");

//asetetaan kahva asynkroniseen tehtavaan
        gCallbackHandle = hdScheduleAsynchronous(JousivoimanLaskeminen, 0,
HD_MAX_SCHEDULER_PRIORITY);


//sallitaan voiman tuotto
  hdEnable(HD_FORCE_OUTPUT);

//kaynnistetaan aikatauluttaja
  hdStartScheduler();

  if (HD_DEVICE_ERROR(error = hdGetError()))
  {
        hduPrintError(stderr, &error, "Aikatauluttajan kaynnistaminen epaonnistui\n");

        hdDisableDevice(hHD);                   //kytketaan pois haptiikkalaite
        glfwTerminate();                        //sammutetaan GLFW
        getch();                                        //odotetaan napin painallusta
        return 0;
    }

 printf("Voit muutta jousen jaykkyytta nuolinappaimilla (ylos + ja alas -)\n\n");
while(1)
{
  if (!hdWaitForCompletion(gCallbackHandle, HD_WAIT_CHECK_STATUS))
    {
      fprintf(stderr, "\nThe main scheduler callback has exited\n");
      fprintf(stderr, "\n Paina nappia lopettaaksesi \n");
      getch();
      hdStopScheduler();
      hdUnschedule(gCallbackHandle);
      hdDisableDevice(hHD);
      glfwTerminate();
      return 0;
    }


  if(glfwGetKey(GLFW_KEY_UP) == GLFW_PRESS)
    {
      Jousi +=  muutos;

      //jos jaykkyys ylittaa laitteen maksimi jaykkyyden sijoitetaan maksimijaykkyys
```

```
            if (Jousi > MaxJaykkyys) Jousi = MaxJaykkyys;
            //tulostetaan jousivakio
            printf("Jousivakio %.2f N/mm \n", Jousi);
     //Kutsutaan aikatauluttajaa joka muuttaa jousivakion
                                        hdScheduleSynchronous(JousivoimanAsettaminen,        &Jousi,
HD_DEFAULT_SCHEDULER_PRIORITY);

    }
  if(glfwGetKey(GLFW_KEY_DOWN) == GLFW_PRESS)
    {
      Jousi -= muutos;

     //ei anneta menna negatiiviseksi, ettei jousivoima muuta suuntaa, ei ala vahvistamaan
            if(Jousi < 0.01) Jousi = 0.01;
            //tulostetaan jousivakio
            printf("Jousivakio %.2f N/mm \n", Jousi);
     //Kutsutaan aikatauluttajaa joka muuttaa jousivakion
                                        hdScheduleSynchronous(JousivoimanAsettaminen,        &Jousi,
HD_DEFAULT_SCHEDULER_PRIORITY);

    }

  if(glfwGetKey('Q') == GLFW_PRESS)
    {
      hdStopScheduler();
      hdUnschedule(gCallbackHandle);
      hdDisableDevice(hHD);
      glfwTerminate();
      return 0;
    }
  glfwPollEvents();

  //viive ettei jousivakiota kasvateta liian nopeasti
  glfwSleep(0.2);


 }
 return 0;
}


//aikatauluttaja
HDCallbackCode HDCALLBACK JousivoimanLaskeminen(void *p_Kayttajan_tieto)
{
    HDErrorInfo error;
    static hduVector3Dd ankkuri;
    hduVector3Dd paikka;
    hduVector3Dd voima;


          voima[0]=0;
          voima[1]=0;
          voima[2]=0;
```

```
        ankkuri[0]=0;
        ankkuri[1]=0;
        ankkuri[2]=0;


    hdBeginFrame(hdGetCurrentDevice());

    hdGetDoublev(HD_CURRENT_POSITION, paikka);

    hduVecSubtract(voima, ankkuri, paikka);

    hduVecScaleInPlace(voima, Jousivakio);

    hdSetDoublev(HD_CURRENT_FORCE, voima);

    hdEndFrame(hdGetCurrentDevice());

    //tarkastetaan onko tapahtunut virheita
    if (HD_DEVICE_ERROR(error = hdGetError()))
      {
       if (hduIsForceError(&error))
            {
            printf("Voiman asettaminen epaonnistui");
            }
       else if (hduIsSchedulerError(&error))
            {
            return HD_CALLBACK_DONE;
            }
      }
    return HD_CALLBACK_CONTINUE;
}


HDCallbackCode HDCALLBACK JousivoimanAsettaminen(void *p_Kayttajan_tieto)
{
    HDdouble *p_Jaykkyys = (HDdouble *) p_Kayttajan_tieto
    //sijoitetaan uusi jousivakio
    Jousivakio = *p_Jaykkyys;
          //palautetaan tieto, etta tehtava suoritettu
    return HD_CALLBACK_DONE;
}
```