Felhasználói felület kialakítása Microsoft Kinect 3D érzékelő segítségével

Pál Gábor
Konzulens: Dr. Vajda Ferenc

Irányítástechnika és Informatika Tanszék
Villamosmérnöki és Informatikai Kar
Budapesti Műszaki és Gazdaságtudományi Egyetem

2012.12.02.

Tartalomjegyzék

1. Bevezetés

1.1. Natural User Interfaces[1]

A Natural User Interfaces (gyakorta Natural Interfaces néven találkozhatunk vele) egy olyan ember-gép interakciót ír le, amelyben a gép vezérléséhez vagy irányításához az ember semmilyen külső eszköz segítségét nem veszi igénybe, csak a saját adottságaival visz át információt a gép felé. Ezek a saját adottságok legtöbbször mozdulatok, valamint szóban kiadott utasítások. Ezáltal feleslegessé válnak az olyan jól megszokott beviteli eszközök, mint a billentyűzet és az egér.

Néhány példa a Natural User Interfaces mindennapi használatából:

- Hangfelismerés
- Kézmozdulat felismerés: néhány előre definiált kézmozdulattal vezérelhetjük a rendszert.
- Mozgás felismerés: a szenzor az egész testet figyeli, a végtagok mozgatásával vezérelhetjük a rendszert. Legtöbbször játékok irányítására használják.

1.2. Szenzorok

Természetéből kifolyólag a Natural User Interfaces technológiák használatához szükségünk van egy vagy több szenzorra, amely figyeli a mozdulatainkat vagy felveszi az általunk kiadott hangokat. Ezeket a szenzorokat a felhasználás helye szerinti két nagy csoportba oszthatjuk.

1.2.1. Számítógéphez készített

A két legelterjedtebb, számítógéphez készített szenzor a Microsoft által kiadott (a PrimeSense cég fejlesztésében készült) „Kinect for Windows” és az Asus által forgalmazott „Xtion PRO” és „Xtion PRO LIVE”. Ezekben közös, hogy USB interfészen át kommunikálnak a számítógéppel, és legfőképp programozási, alkalmazásfejlesztési célokkal hozták létre őket. Egyelőre magas áruk és viszonylagos gyenge támogatottságuk miatt nem terjedtek el eléggé a piacon. A Microsoft ezen azzal próbál segíteni, hogy a Windows 8 operációs rendszerébe beépíti a „Kinect for Windows” támogatását, ezáltal a Metro felület vezérelhető lesz ezzel az eszközzel.

1.2.2. Más eszközhöz készített

Egyéb eszközökhöz készített szenzorok között döntő többségében különböző játékkonzolokhoz kifejlesztett eszközöket találunk. Ezeket kizárólagosan játékok vezérlésére hozták létre, és legtöbbjüknél kifejezetten tiltott bármely egyéb felhasználási terület, mint az adott konzolon futó játékok vezérlése. A három legelterjedtebb ilyen szenzor a Microsoft által az Xbox konzolhoz kiadott (és szintén a PrimeSense fejlesztésében készült) „Kinect for Xbox” (a szakirodalomban „Kinect”-ként hivatkozva mindig ezt az eszközt értik, és nem a Windows operációs rendszerre készült párját), a Sony cég által a PlayStation konzolhoz készített „Move motion controller” és a Nintendo Wii eszközéhez kiadott kontroller. Ez utóbbi kettőről nem állítható teljes mértékben, hogy a Natural User Interfaces technológiát

[1] http://openni.org/Documentation/ProgrammerGuide.html

használná, ugyanis a Move és a Wii esetében is a kezünkben kell tartanunk egy érzékelőt, amely a kezünk aktuális helyzetéről gyűjt és küld információkat a konzolnak. Kijelenthető tehát, hogy a piacon jelenleg két elterjedt szenzor van, amely minden tekintetben kielégíti a Natural User Interfaces követelményeit, ez pedig az Asus Xtion PRO és a Microsoft Kinect.

2. Kinect hardver

Ebben a fejezetben részletesen bemutatom a Kinect hardver történetét, felépítését, az általa nyújtott szolgáltatásokat és a különbségeket a két különböző platformra (Xbox és Windows) készült változatok között.

2.1. Áttekintés és történelem[2][3][4][5][6]

Az eredetileg Project Natal néven futó technológiát 2010 harmadik negyedévében mutatta be a Microsoft (ekkor még csak Xbox-ra). A bemutatott szenzor képes volt felismerni a mozgás-, a hang- és a mélységadatokat. Alig 3 nappal a megjelenés után (ekkor Európában hivatalosan még meg sem jelent) már sikerült feltörni a Kinectet, aminek a felhasználó követésére használatos motorját (tilt motor) sikerült PC-ről vezérelni. Ebben ösztönzőleg hathatott egy amerikai hardver cég 1000, majd később 2000 dolláros ajánlata, amit a Kinectet elsőként feltörőnek ajánlott fel. A Microsoft ekkor még ellenezte a Kinect Xbox játékoktól független felhasználását, és jogi lépéseket helyezett kilátásba az eszköz bármely más célra történő felhasználása esetén. Bejelentették továbbá, hogy olyan változtatásokat építenek az eszközbe, amelyek lehetetlenné teszik az Xbox-tól eltérő használatot. Időközben a Kinect bekerült a Guinness világrekordok könyvébe is, ugyanis minden addigi eladási rekordot megdöntött: két hónap alatt nem kevesebb, mint 8 millió darabot adtak el belőle világszerte. Ez naponta több mint 133.000 db-ot jelent. Nem meglepő, hogy a termék terjedésével párhuzamosan elkezdtek elterjedni a PC-s felhasználást elősegítő különböző megoldások, azaz tört SDK-k is. Ezek többé-kevésbé támogatták a Kinect funkcióinak kihasználását, viszont legtöbbjük elég alacsony szintű adatokat továbbított csak a felsőbb szintű alkalmazások felé.

A kezdeti elhatárolódás után (valószínűleg a tört SDK-k terjedését látva) a Microsoft 2011. június 16-án kiadta a Kinect SDK béta verzióját, amivel lehetővé vált az eszköz Windows 7 operációs rendszer alóli programozása Visual Basic, C++ és C# nyelven is. Ezzel a lépéssel a Microsoft feladta addigi nézeteit, hogy kizárólag játékra lehet felhasználni a Kinectet.

2012. január 10-én a CES konferencián (Consumer Electronics Association, az elektronikai termékeket gyártó cégek minden év januárjában megrendezett konferenciája, melyen az év során várható újdonságokat mutatják be) a Microsoft bejelenti a Kinect for Windows

[2] http://pcforum.hu/hirek/12381/Feltortek+a+Microsoft+Kinectet.html
[3] http://www.hwsw.hu/hirek/46889/microsoft-kinect-sdk-fejlesztes.html
[4] http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx
[5] http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/31/kinect-for-windows-is-now-available.aspx
[6] http://prohardver.hu/hir/kinect_for_windows_vegre_megjelent_microsoft.html

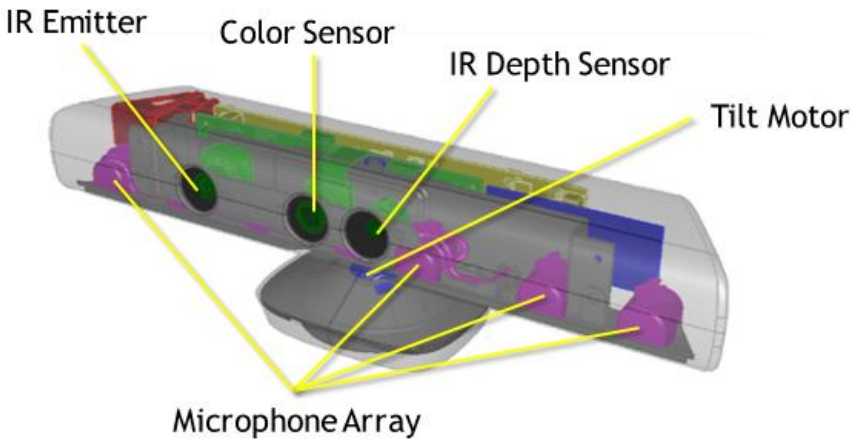
elkészültét, és forgalmazásának beindítását 250 dolláros áron. Ez az ár jóval drágább, mint az Xbox-os verzió ára, ugyanis itt a hardver árának egy részét nem tudják finanszírozni a játékok eladásából befolyó bevételből. A hardver megjelenésével párhuzamosan megjelent a hozzá tartozó fejlesztői készlet (SDK) első teljes verziója is. A működéshez egy erősebb 2 magos processzorra volt szükség, továbbá a Visual Studio 2010 valamely verziójára, valamint a .NET Framework 4.0-ra. 2012 májusában jelent meg az SDK 1.5-ös verziója, amely sok újdonságot jelentett az addigi változatokhoz képest. Megjelent a felhasználók arcának követése, valamint bekerült az SDK-ba a mozdulatsorok rögzítését végző, majd később azt lejátszani tudó szoftvermodul is.

Jelenleg (2010. 12. 02.) Magyarországon hivatalosan csak a Kinect for Xbox kapható, de a hozzá járó Xbox-os játékból is adódóan ezt egyértelműen csak játék célra szánja a Microsoft, amelyet a termékhez tartozó felhasználási és szerződési feltételekben szigorúan ki is köt. Többszöri ígéret és dátummódosítás után sem kapható még a Kinect for Windows hivatalos magyar viszonteladónál. A Microsoft szakmai blogja alapján a legutóbbi dátum a megjelenésre 2012. október 8. volt, viszont a termék ekkor sem került a boltokba. Jelenleg nincs információ arról, hogy tervezi-e kiadni a Microsoft hazánkban is a Kinect for Windows terméket, és ha igen, akkor mikor.

2.2. Kinect for Xbox[7][8]

1. ábra: Kinect for Xbox

[7] http://msdn.microsoft.com/en-us/library/jj131033.aspx
[8] Andrew Davison – Kinect Open Sourse Programming Secrets: 2. oldal



2. ábra: A Kinect felépítése

A Kinect for Xbox a 2. ábrán található modulokból épül fel. Talán a legfontosabb szenzorjai az IR Emitter és az IR Depth Sensor. Az IR Emitter infravörös tartományú fényt bocsájt ki, amely visszaverődve a különböző tárgyakról, és az emberi testről az IR Depth Sensor-ban (ez egy monokróm CMOS infravörös szenzor) kerül feldolgozásra. Ebből a szenzorból lehet kinyerni a mélységi adatokat, ami talán a leggyakrabban használt része a Kinectnek. Az IR Depth Sensor a képpontokat távolsági adatokká konvertálja át, így ezek az adatok később mélységi mátrixként (vagy mélységi térképként) is felhasználhatóak. A szenzor megbízható érzékelési tartománya 50 cm-től 3 méterig terjed (a valós, ámde néha bizonytalan érzékelési tartomány 50 cm-től kb. 5 méterig terjed). A szenzor érzékelési pontossága mélységi adatokra (z tengely) 1 cm, míg szélességi és magassági adatokra (x és y tengely) milliméterekben mérhető. Az érzékelési pontosság a távolság változásával nem lineáris arányban változik, erről egy bővebb cikk itt olvasható: http://mathnathan.com/2011/02/depthvsdistance/. Az IR Depth Sensor által továbbított adatfolyam 640x480 pixel felbontású, és 11 bites távolsági adatokat tartalmaz, ezáltal 2048 db különböző távolsági adat ábrázolható. Az adatfolyam maximális frekvenciája 30 Hertz, de ez szoftveresen konfigurálható. Az IR Depth Sensor látószöge vertikálisan (x tengely) 43°, míg horizontálisan (y tengely) 57°. Ez utóbbi a beépített motor segítségével tovább javítható.

Található az eszközben egy szín szenzor (Color Sensor) is, amely egy webkamera lehetőségeit nyújtja a felhasználó számára. Az IR Dept Sensor-hoz hasonlóan ez is VGA-nak megfelelő 640x480 pixel méretű képeket vagy videókat készít 30 Hertzes frissítési frekvencia mellett.

A 4 darab mikrofon (Microphone Array) feladata a hangok rögzítése és továbbítása a konzol vagy a számítógép felé. A mikrofonoktól érkező hangfolyamok egyesével is feldolgozhatóak, ennek segítségével könnyen megállapítható, hogy a hangforrás a szenzorhoz képest milyen irányban található. A mikrofonok 16 KHz-es adatmintákat továbbítanak PCM kódolással (Pulse Code Modulation). A beszédfelismerő algoritmusok implementálása során nagy segítség lehet a hardverbe beépített automatikus zajszűrés (Ambient Noise Suppression).

Belekerült még a hardverbe egy vertikális mozgást lehetővé tévő motor is (Tilt Motor), amely az eszközt a talapzatához viszonyítva ±27°-kal képes mozgatni.

Az IR Depth Sensor és a Color Sensor fent említett 640x480-as felbontása helyett szoftveresen konfigurálható 1280x1024-es felbontás is, viszont itt csak 10 Hertzes frissítési

frekvencia érhető el. Ez hasznos lehet, ha nagyobb felbontású képekre vagy mélységi adatokra van szükségünk, viszont ilyen frissítési frekvenciával nem lehet használható mozgóképet előállítani.

Található még az eszközben egy, az ábrán nem látható status LED, amely villogással jelzi, ha az eszköz áramellátás alatt van, és konstans módon világít, ha a szenzor használat alatt is van (ez azt jelenti, hogy a Kinect valamely szenzora éppen adatokat továbbít a konzol vagy a PC felé). További beépített szenzor a gyorsulásmérő is, viszont a Kinectre épülő alkalmazások tekintetében ez elég ritkán használt.

Ahogy az 1. ábrán is látható, a Kinectet nem csak a számítógép vagy az Xbox USB portjához kell csatlakoztatni, de külső áramellátásra is szüksége van. A Kinect maximális fogyasztása 12 watt, és ez jóval több az USB szabványban rögzített 2.5 wattos maximális áramfelvételnél, amely egy USB portra kötött eszköztől elvárható lenne.

2.3. Kinect for Windows[9]

3. ábra: Kinect for Windows

A Kinect for Windows termékben minden megtalálható, ami az Xbox-hoz készült változatában is, ezért ezen tulajdonságok taglalására nem térnék ki részletesen. Nézzük inkább, hogy mivel nyújt többet ez az eszköz, mint az elődje: mindenképp magasabb árkategóriát képvisel ez a termék, ugyanis az Xbox-os verzió 150 dollárjához képest ezt 250 dollárért árulják kiskereskedelmi forgalomban. Ezért az árért természetesen jobb képességű hardvert, és pluszszolgáltatásokat kapunk. A Kinect for Windows használva a „near mode" technológiát már 40 cm-ről is lehetővé teszi a mozdulatok megbízható azonosítását, szemben az Xbox-os változat által biztosított kb. 50-80 cm helyett. Lehetőség van egyidejűleg 4 db ilyen eszköz csatlakoztatására is a PC-hez, amelyek képesek javítani egymás pontosságát, vagy a tér különböző részeit pásztázva képesek minél átfogóbb képet alkotni. A Skeletal Tracking (az eszköz előtt álló felhasználóból egy csontvázszerű képet készít, amelyen néhány pont változását követi) technológia fejlesztése is megtörtént, most már szoftveresen állíthatja a felhasználó, hogy az eszköz előtt álló emberek közül melyiket kövesse a szenzor (eddig mindig az először érzékelt embert kezdte el követni). Ezen kívül fejlesztettek még a

[9] http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/31/kinect-for-windows-is-now-available.aspx

zajszűrésen, valamint a hozzá tartozó SDK-ból is új változatot adtak ki. Ebben több és részletesebb példaprogram található, melyekkel a Kinect for Windows minden szenzorát ki lehet próbálni.

2.4. Különbségek a Kinect for Xbox és a Kinect for Windows között

Az alábbi táblázat tartalmazza a főbb különbségeket a két eszköz között.

4. ábra: Kinect for Xbox és Kinect for Xbox dobozai

	Kinect for Xbox	Kinect for Windows
Ár	150$	250$ (150$ a programra regisztrált egyetemek számára)
Legkisebb távolság a mélységet érzékelő szenzortól	50-80 cm	40 cm
Mit tartalmaz a doboz (a Kinect mellett)	Egy Kinect Advantures nevű Xbox játékot	Néhány instrukciót az SDK-hoz, és a különböző szoftverek letöltéséhez
Célközönség	Játékosok	Asztali alkalmazásokat fejlesztők (nagyrészt vállalatok és kutatók)
USB kábel hossza	Hosszú (a Kinect és a Xbox nagyobb távolsága miatt)	Rövid (a PC általában elég közel van a Kinecthez)
USB csatlakozó	Xbox360 szerinti „narancssárga” csatlakozó + USB kábeltoldás a régebbi Xbox számára	Szabványos USB csatlakozó

3. Kinect szoftver

3.1. Áttekintés

A továbbiakban azokról a szoftverekről lesz szó, amelyek segítségével használatba vehetjük a Kinectet. Legtöbbjüket a Kinect for Xbox változathoz készítették, ebből kifolyólag nem tudják kihasználni a Kinect for Windows nyújtotta új szolgáltatásokat. Csupán a Microsoft

hivatalos SDK-ja biztosítja a mindenkori legújabb hardveres fejlesztésekhez a szoftveres támogatást.

3.2. OpenNI[10]

5. ábra: Az OpenNI hivatalos logója

Az OpenNI (Open Natural Interactions) egy LGPL licence alatt futó (szabadon módosítható és terjeszthető) open source cross-platform API, amely lehetővé teszi a Kinect szenzorjaihoz történő hozzáférést több programozási nyelven is. Cross-platform, mert több platformra is elérhető (Windows, Linux, Android), és az OpenNI keretrendszerben írt kódok függetlenek az alattuk futó operációs rendszertől. Az OpenNI egy egységes keretrendszert nyújt számunkra, amely elrejti az őt használó alkalmazások elől, hogy milyen eszköz fut alatta. Az OpenNI által támogatott beviteli eszközök gyártóinak kötelező implementálni bizonyos szolgáltatásokat, amelyeket később egységes függvényekkel érhetünk el. Például ha egy RGB kamera által szolgáltatott képet szeretnénk megkapni, akkor nem kell foglalkoznunk az RGB kamera pontos specifikációjával, csupán meghívni az OpenNI által biztosított, erre szolgáló függvényt. Természetesen így a kódunk nem fog függeni a bemeneti adatokat nyújtó hardvertől, azaz ha egy másik (OpenNI által támogatott) RGB kamerára cseréljük ki az eddigi kameránkat, akkor a már megírt összes ezt használó alkalmazás zavartalanul használható tovább.

Az OpenNI jelenleg C, C++, Java és C# nyelveken érhető el. További információ az OpenNI-ról ezen dokumentum 4. fejezetében, továbbá az OpenNI hivatalos oldalán (http://www.openni.org/) található.

3.3. OpenKinect[11]

6. ábra: Az OpenKinect hivatalos logója

Az OpenKinect egy nyílt közösség, amely céljául tűzte ki, hogy a legjobb, legsokoldalúbb Kinect alkalmazásokat fejlesszék. A projekt eredetileg Hector Martin kezdeményezésére indult, aki már a megjelenést követő néhány napban reverse engineering módszerrel (kódvisszafejtés) visszafejtette a Kinect driver forráskódját, ezzel elnyerve a 2.1. pontban

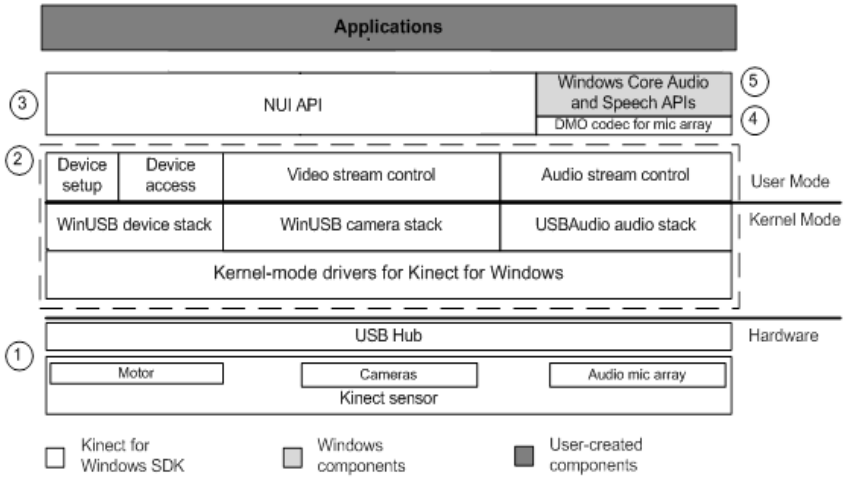
[10] http://openni.org/Documentation/ProgrammerGuide.html
[11] http://openkinect.org/wiki/Main_Page

említett jutalmat. Fejlesztői az open source világhoz hasonlóan hobbiból fejlesztik az OpenKinectet, jelenleg több mint 2000 lelkes fejlesztő dolgozik szabadidejében a projekten. Az OpenNI-jal ellentétben az OpenKinect (a nevéből eredően) csak a Kinectet támogatja, azaz nem használható semmilyen egyéb beviteli eszközzel.

Az OpenKinecten belül jelenleg a legelterjedtebb Kinect driver a Libfreenect. Az OpenNI-hoz hasonlóan a Libfreenect is cross-platform, open source, GPL2 licence alatt futó szoftver, amely jelenleg Windows, Linux és Mac operációs rendszereken érhető el. A Libfreenect által támogatott programozási nyelvek: C, C++, C#, Java, Python. Függvényei alacsonyabb szintűek (kevesebb absztrakciós réteget tartalmaznak), mint az OpenNI hasonló függvényei, viszont gyorsabbak annál, kihasználva azt a tulajdonságot, hogy kevesebb generális megoldásra van szükség, mivel csak a Kinectet kell támogatni. Az OpenNI-jal legtöbbször használt alacsony szintű driverrel (SensorKinect) ellentétben a Libfreenect támogatja a beépített motor, a gyorsulásmérő és a status LED használatát is.

További információ az OpenKinectről és a Libfreenectről:
http://openkinect.org/wiki/Main_Page

3.4. Windows SDK[12]



7. ábra: A Windows SDK felépítése

Nevéből eredően a Windows SDK-t a Microsoft adta ki először béta verzióban a Kinect for Xbox termékhez, majd az első teljes verzió a Kinect for Windows megjelenésével párhuzamosan érkezett. A Microsoft üzletpolitikájából eredően az SDK nem nyílt forráskódú, és csak Windowsra érhető el, de ott legalább ingyenes a felhasználása (nem üzleti célokra). Mivel az SDK-t ugyanúgy a Windows adja ki, mint a Kinectet, ezért mindig ebben jelenik meg először a Kinectre érkező frissítések támogatása. További előnye, hogy rendkívül jól dokumentált, mind a programozói kézikönyv, mind a hozzá tartozó példák részletesen, és jól bemutatják a programozási felületet.

További információ a Windows SDK-ról:
http://msdn.microsoft.com/en-us/library/hh855347.aspx

[12] http://msdn.microsoft.com/en-us/library/hh855347.aspx

3.5. CL NUI[13]

A Code Laboratories Natural User Interfaces nevű szoftverével is lehetőség nyílik a Kinecttől érkező adatok feldolgozására. A Code Laboratories egy kutatás-fejlesztési csapat, amely valósidejű interaktív rendszerekhez készít drivereket és alkalmazásokat. A NUI nevű projektjükön kívül több más dolgot is fejlesztenek. Nem teljesen helyénvaló összehasonlítani a CL NUI-t a fenti három szoftverrel, ugyanis míg az előző három főként a Kinect programozására és minden lehetőségének kihasználásra készült, addig a CL NUI ennél kevesebbet nyújt, és inkább csak kapcsolatot teremt a Kinect és a PC között. Előnye, hogy egyszerű grafikus felhasználói felülettel rendelkezik, és néhány gombnyomás után már láthatjuk is a Kinect által gyűjtött adatokat, valamint mozgathatjuk a hozzá tartozó motort. Természetesen lehetőséget ad saját programok írására is, viszont sokkal alacsonyabb szintről kell elkezdenünk a fejlesztést, mint a fent említett három szoftver bármelyikénél.

További információ a CL NUI-ről: http://codelaboratories.com/nui

4. OpenNI

4.1. Áttekintés

A 3.2. pontban már röviden bemutatott OpenNI nagyon fontos tulajdonsága, hogy nem kifejezetten a Kinecthez készült, hanem támogat minden olyan beviteli hardvert, ami megfelel az OpenNI által támasztott követelményeknek. Ez azt jelenti, hogy az OpenNI magában nem tartalmaz sem drivereket sem a szenzorok által küldött adatokat feldolgozó függvényeket. Ezeket a feladatokat a 4.3. pontban tárgyalt beépülő modulok végzik. Az OpenNI által ellátott három legfontosabb feladat:

- Biztosítani a beépülő modulok közötti gyors és hatékony kommunikációt.
- Egységes, jól definiált interfészeket nyújtani a beépülő modulok számára.
- Jól definiált és megfelelően dokumentált interfészeket nyújtani az OpenNI-t használó alkalmazások számára.

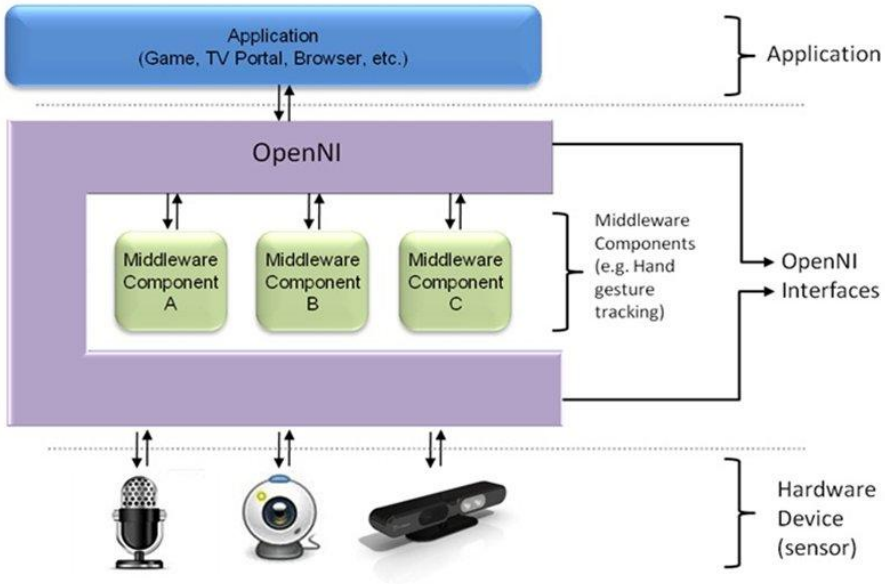
Nagy előnye az OpenNI-nak, hogy nincs megkötve a beépülő modulok száma, és az általuk támogatott funkcionalitások, így akár ugyanazt a funkcionalitást nyújtó, más gyártótól származó modult is használhatunk az OpenNI vagy az általunk írt programok módosítása nélkül. Ezek a modulok egymásra is épülhetnek, azaz az egyik modul kimenetét használhatja egy másik modul bemenetként. A későbbiekben erre is láthatunk majd példát.

Jelen pillanatban az OpenNI nem támogatja a Kinectbe épített motor, gyorsulásmérő és a status LED használatát, viszont lehetőséget biztosít az USB interfész programozására, azaz alacsony szintű eszközökkel az OpenNI-on keresztül is elérhetőek ezek a szolgáltatások. Mivel ezek használata meglehetősen kényelmetlen, ezért a jelenlegi OpenNI-ra épülő alkalmazásokban a fenti három funkció közül szinte egyiket sem találhatjuk meg.

[13] http://codelaboratories.com/nui

Felvetődhet a kérdés, hogy ha mindenképp szükség van a fenti funkciók valamelyikére, akkor érdemes-e várni, amíg az OpenNI támogatni fogja őket, vagy megpróbálkozni az alacsony szintű megoldások valamelyikével. Sajnos több fórum egybehangzó véleménye alapján nem várható, hogy az OpenNI a közeljövőben támogassa ezeket a funkciókat, ugyanis ezek eléggé Kinect-specifikus szolgáltatások, amik jellemzően nem találhatóak meg, vagy teljesen eltérően viselkednek a többi OpenNI által támogatott hardverben. A támogatásuk talán akkor kerülhet be az OpenNI-ba, ha piacra kerül több, az OpenNI által támogatott hardver is, amely hasonló megoldásokat alkalmaz a fenti funkciók implementálására, mint ami a Kinectben is megtalálható.

4.2. Architektúrális felépítés[14]



8. ábra: Az OpenNI architektúra felépítése

A 8. ábrán látható az OpenNI felépítése. A 3 réteg főbb feladatai:

- Applikáció réteg: Az OpenNI szolgáltatásait használó alkalmazásokat reprezentálja, amelyeknek minden, az OpenNI által támogatott platformon ugyanúgy kell viselkedniük.
- OpenNI Interfaces réteg: Az OpenNI-t megvalósító réteg. Kapcsolatot biztosít az applikációs és a hardver réteg között, valamint ezen rétegen belül futnak a különböző beépülő modulok, amelyek a hardver rétegtől kapott adatokat dolgozzák fel.
- Hardver réteg: A különböző szenzorokat megvalósító hardvereszközök, amelyek mindenfajta feldolgozás nélküli, úgynevezett nyers (raw) adatot továbbítanak az OpenNI felé.

4.3. Beépülő modulok

A 8. ábrán látható OpenNI Interfaces rétegbe különböző modulokat (middleware components) építhetünk be. Ezek nagyrészt hardver specifikus modulok, amelyek elrejtik az OpenNI elől a

[14] http://openni.org/Documentation/ProgrammerGuide.html

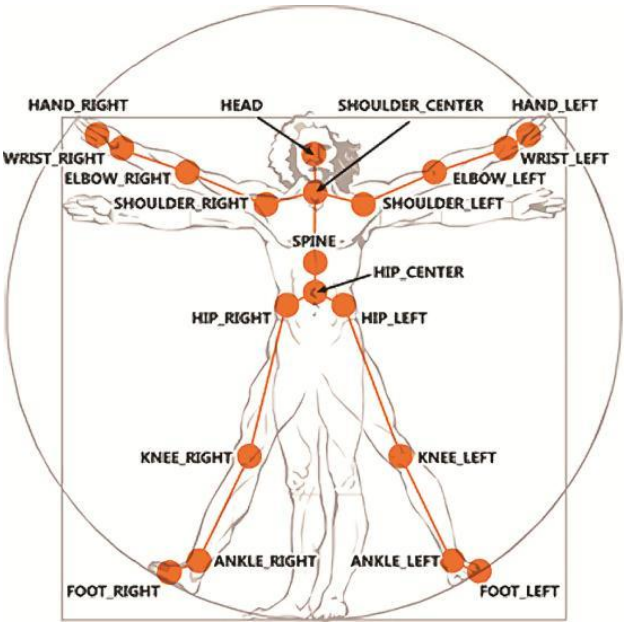
hardver egyes tulajdonságait. Léteznek nem hardver specifikus modulok is, ezek tipikusan egy vagy több hardver specifikus modul kimenő adatait dolgozzák fel. A következőkben a Kinect használatához szükséges modulokat mutatom be.

4.3.1. NITE[15][16]

A NITE a PrimeSense cég által fejlesztett middleware komponens az OpenNI-hoz. Ugyanez a cég fejlesztette ki a Microsoft számára a Kinect hardvert, így a megfelelő tudás birtokában ők készítették el a legjobban használható magasabb szintű modult az OpenNI-hoz. A NITE az alábbi szolgáltatásokat nyújtja az alkalmazásfejlesztők számára:

- Teljes test analízis (full body analysis middleware): a teljes testet analizálja, és előállít néhány, az applikációs rétegben felhasználandó adatot. Ilyenek lehetnek pl. a testen azonosított pontok, vagy az ezekből a pontokból összeállított csontváz.
- Kéz analízis (hand point analysis middleware): A kéz pontjait analizálja, és előállít, majd követ egy pontot a tenyér közepén. Felfelé már csak ennek a pontnak az adatait propagálja.
- Gesztus analízis (gesture detection middleware): Végtag mozdulatokkal különböző, előre megadott mintákat lehet leírni (pl. integetés, kéz elmozdítása a testtől egy bizonyos szögben), amelyeket a NITE azonosítani tud. Felfelé már csak ezeknek a gesztusoknak a bekövetkeztét továbbítja.
- Kép analízis (scene analyzer middleware): Analizálja, és elkülöníti az előtérben található személyeket a háttérben található tárgyaktól, valamint azonosítja az előtérben található személyek egyes testrészeit, és különböző színekkel jelöli az egyes embereket. Ajánlott homogén hátteret ezen szolgáltatáshoz, ugyanis a követendő személlyel azonos mélységben található tárgyak (pl. szék vagy asztal) megzavarhatják a komponens működését. Előfordul néhányszor, hogy egy szék vagy asztallábat sikerül a felhasználó lábának azonosítani. Ezt viszonylag gyorsan korrigálja a NITE, viszont a valós idejű használatban gondot okozhat.

[15] http://www.primesense.com/nite
[16] http://openni.org/Documentation/ProgrammerGuide.html



9. ábra: A NITE teljes test analízise által előállított pontok, és azok nevei

4.3.2. SensorKinect[17]

A SensorKinect nevű hardver komponenst használhatjuk a Kinect alacsony szintű, nyers adatainak feldolgozására. Nem tartalmaz kifejezetten az applikációs szint számára szolgáló interfészt, nagyrészt a NITE middleware komponenseivel történő kommunikációra tervezték. Ebből kifolyólag nem rendelkezik kiterjedt dokumentációval, csak a forráskódhoz mellékelt readme fájlokból, illetve a forráskódban szereplő kommentekből lehet tájékozódni a pontos működéséről. Mivel az OpenNI jelenleg nem támogatja a motor, a gyorsulásmérő és a status LED használatát, ezért ezek a SensorKinect implementációjából is kimaradtak.

Viszonylag ritkán történik fejlesztés rá, ugyanis a Kinect hardver sem gyakran változik, míg magasabb szintű feladatot a SensorKinect nem végez, mint a nyers adatok megfelelő feldolgozását, és továbbítását a programozó által emészthető formátumban. A legutóbb kiadott verzió már támogatja a Kinect for Windows használatát is, viszont csak a Kinect for Xbox-ban már megszokott funkciók érhetők el vele.

További információ és a forráskód az alábbi GitHub elérhetőségen található:

https://github.com/avin2/SensorKinect/commits/unstable

4.4. Production node-ok és production chain-ek[18]

Érhető és feldolgozható 3D adatot előállítani összetett feladat, amely komplex architektúrát igényel. A fent említettek szerint az OpenNI ezt a folyamatot egymásra épülő modulokkal oldja meg, amelyek lehetnek hardver vagy middleware komponensek. Egy tipikus gesztus analízis például a következők szerint zajlik: a Kinect hardver a nyers adatot (mélység információkat) továbbítja egy hardver modul felé, amely érthető mélységi információkat állít elő a nyers adatból, amit továbbít egy teljes test analízist végző middleware komponens felé. Ez a middleware komponens analizálja a test egyes pontjait, majd ezeket az információkat

[17] http://openni.org/Documentation/ProgrammerGuide.html
[18] http://openni.org/Documentation/ProgrammerGuide.html

továbbítja egy gesztus analizáló middleware komponens felé, amely a kapott adatokból megállapítja, hogy megtörtént-e a kívánt gesztus, vagy sem, majd erről tudatja az ő szolgáltatásait (az OpenNI-on keresztül) használó alkalmazást. Az alkalmazáskészítőnek így könnyű dolga van, ugyanis nem kell a fent leírt folyamatot implementálnia az alkalmazásában, elég csak megfelelően paraméterezett függvényhívások visszatérési értékeit ellenőriznie.

A fent leírt folyamatban production node-oknak nevezzük a hardver vagy middleware komponensek egy halmazát, amelyek részt vesznek a kapott adatok feldolgozásában, illetve új adatok előállításában. Minden production node-nak jól definiált feladatkörrel kell rendelkeznie, valamint szolgáltatásokat kell nyújtania az őt igénybe vevő egyéb production node-oknak vagy alkalmazásoknak. Ezek a production node-ok az OpenNI alapkoncepcióját képezik, ugyanis ők nyújtanak publikus interfészeket az alkalmazások számára, és ezeket felhasználva tudunk könnyen feldolgozható információt kinyerni a Kinectből. Generálisan a production node-okról elmondható, hogy mindegyikük egy különálló egység, amely alacsonyabb szintű production node-ok adatait dolgozza fel, és az általa előállított adatokat magasabb szintű production node-oknak továbbítja. Tipikusan jellemző az OpenNI-t használó alkalmazásokra, hogy több production node-ot is használnak az általuk szükséges adat előállításához, habár konkrétan csak 1 production node-dal vannak kapcsolatban. A production node-ok ilyen típusú láncait production chain-eknek hívjuk. Ha több, ugyanolyan szolgáltatást biztosító modul is jelen van a rendszerben, akkor az alkalmazás készítője szabadon választhat, hogy ezek közül melyeket szeretné használni (azaz elképzelhető, hogy ugyanannak az eredménynek az eléréséhez több production chain is használható). Minden egyes OpenNI-ra épülő alkalmazás inicializálása során szükség van a megfelelő production chain kiválasztására, még akkor is, ha csak egy lehetséges production chain létezik a rendszerben.

A production node-oknak az architektúrában elfoglalt helyük szerint két fő csoportja létezik, hasonlóan, mint a beépülő moduloknak:

- Hardver szintű production node-ok: kommunikálnak a hardverrel, és a tőle kapott adatokat feldolgozva továbbítják egy vagy több middleware szintű production node-nak.
- Middleware szintű production node-ok: egy hasonló szintű production node-tól, vagy hardver szintű production node-tól kapott adatokat feldolgozva továbbítják egy másik, szintén middleware szintű production node-nak, vagy magának az alkalmazásnak.

A production node-okat felhasználási területük szerint 3 fő csoportba oszthatjuk:

- Generátorok: olyan production node-ok, amelyek új adatot generálnak a felettük lévő production node-ok vagy applikációk számára az általuk kapott adatok alapján.
- Analizátorok: olyan production node-ok, amelyek nem generálnak új adatot a felettük lévő production node-ok vagy applikációk számára, csupán az általuk kapott adatok alapján hozott döntéseiket közlik a felettük lévő réteggel.
- Rögzítő, lejátszó: létezik egy rögzítő (recorder) nevű production node, amellyel rögzíteni tudunk eseményeket (mozgás, hang), amelyek a Kinect előtt történtek. Ezeket az eseményeket később a lejátszó (player) nevű production node segítségével

játszhatjuk le. Ezen funkció segítségével könnyebbé válik az alkalmazások tesztelése, ugyanis nem kell minden egyes alkalommal ugyanazt a mozdulatsort vagy hangeffektust produkálni, elég csupán a lejátszó production node-tól kapott információt feldolgozni.

4.5. Capabilities[19]

Akármennyire is törekszik az OpenNI egy egységes framework megvalósítására, a különböző hardverek között mindig is lesznek különbségek. Nem lenne célravezető, ha ezeket a különbségeket olyan indokkal hagynák ki az OpenNI-ból, hogy van olyan hardver, ami nem támogatja őket, így nem lehet generálisan támogatni.

A fenti probléma megoldására találták ki a capability-ket, amelyek különböző production node-ok opcionális tulajdonságait valósítják meg. A hangsúly az opcionálison van, ugyanis a modulok készítőinek nem kötelező implementálniuk az összes capability-t, viszont egy referenciával rendelkezniük kell, hogy melyik capability használható a moduljukkal, és melyik nem. Továbbá minden egyes production node-nak információt kell tudni visszaadnia arra vonatkozólag, hogy a paraméterben megadott capability implementálva van-e benne vagy nem. A 4.4. pontban említett production chain kezdeti kiválasztásakor megadhatjuk, hogy csak olyan production node-okat szeretnénk használni, amelyek bizonyos capability-vel rendelkeznek.

Szerencsére a NITE és a SensorKinect készítői az összes capability-t implementálták a moduljaikba, így a Kinect használatakor az összes létező, OpenNI által biztosított funkcionalitás használható. Ezek a capability-k címszavakban a következők:

- Alternative View
- Cropping
- Frame Sync
- Mirror
- Pose Detection
- Skeleton Generation
- User Position
- Error State of the sensor
- Lock Aware
- Hand Touching FOV edge

4.6. Az OpenNI telepítése

A fentebb tárgyalt összetett architektúra miatt az OpenNI-t nem egyszerű telepíteni, ugyanis minden egyes beépülő modult külön telepíteni kell hozzá a megfelelő sorrendben. Az interneten nem igazán találni egy leírást sem, ami az elejétől a végéig leírná a telepítés részletes menetét, így ezt ebben a pontban megteszem. A telepítés Ubuntu Linux 10.10-es rendszertől felfelé végezhető el az alábbi parancsok segítségével:

[19] http://openni.org/Documentation/ProgrammerGuide.html

Néhány szükséges háttéralkalmazás installálása:

sudo apt-get install git-core cmake freeglut3-dev pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-dev doxygen graphviz mono-complete

Ha nincs a gépen, akkor openjdk installálása:

sudo apt-get install openjdk-7-jdk

OpenNI repository klónozása:

mkdir ~/kinect/bin
cd ~/kinect/bin
git clone https://github.com/OpenNI/OpenNI.git

RedistMaker script futtatása a Platform/Linux könyvtárban, és a kimeneti binárisok installálása (OpenNI-hoz):

cd OpenNI/Platform/Linux/CreateRedist/
chmod +x RedistMaker
./RedistMaker
cd ../Redist/OpenNI-Bin-Dev-Linux-x64-v1.5.2.23/
sudo ./install.sh

Avin2 SensorKinect forráskódjának letöltése:

cd ~/kinect/bin
git clone git://github.com/avin2/SensorKinect.git
cd SensorKinect
git checkout master

RedistMaker script futtatása a Platform/Linux könyvtárban, és a kimeneti binárisok installálása (SensorKinect-hez):

cd Platform/Linux/CreateRedist/
chmod +x RedistMaker
./RedistMaker
cd ../Redist/Sensor-Bin-Linux-x64-v5.1.0.25/
chmod +x install.sh
sudo ./install.sh

Böngésző: http://www.openni.org/Downloads/OpenNIModules.aspx

OpenNI Compliant Middleware Binaries
Stable
PrimeSense NITE Stable Build for Ubuntu 10.10 x64 (64-bit) v1.5.2.23
(értelemszerűen 32 bites gépre a 32 bites tar-t kell letölteni)
Letöltés ide: ~/kinect/bin

cd ~/kinect/bin
tar -xvjpf nite-bin-linux-x64-v1.5.2.21.tar.bz2
rm -f nite-bin-linux-x64-v1.5.2.21.tar.bz2
cd NITE-Bin-Dev-Linux-x64-v1.5.2.21/Data

Erre a lépésre akkor van szükség, ha példaalkalmazásokat szeretnénk futtatni:
Licence módosítása a Sample-Scene.xml-ben, a Sample-Tracking.xml-ben és a Sample-User.xml-ben erről:

<License vendor="PrimeSense" key="insert key here"/>
erre:
<License vendor=”PrimeSense”
key=”0KOIk2JeIBYClPWVnMoRKn5cdY4=”/>

Install script futtatása a NITE könyvtárban:
cd ..
sudo ./install.sh

Ha a Kinect használata során alábbi hiba lépne fel: Failed to set USB interface!
Akkor ezt a parancsot kell használni:
sudo rmmod gspca_kinect

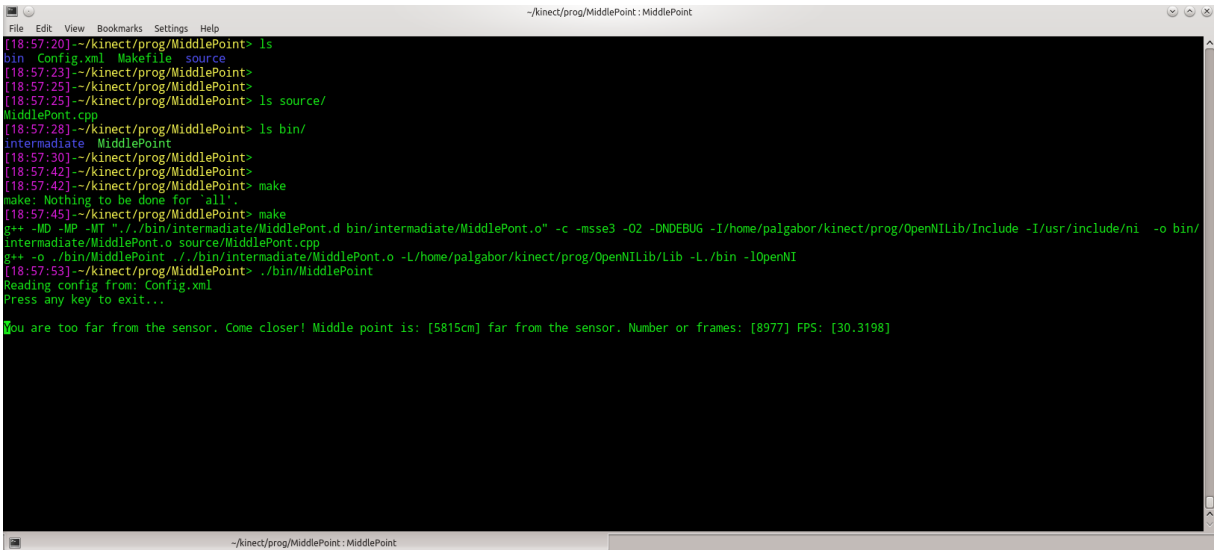
Ezzel a telepítés folyamata lezárult, az OpenNI használatra kész. Található néhány példaprogram is a letöltött fájlok között, ezek az alábbi útvonalon találhatóak:
OpenNI példaprogramok:
cd ~/kinect/bin/OpenNI/Platform/Linux/Bin/x64-Release
NITE példaprogramok:
cd ~/kinect/bin/NITE-Bin-Dev-Linux-x64-v1.5.2.21/Samples/Bin/x64-Release

5. Általam készített programok

Az OpenNI környezettel és a Kinect hardverrel történő alapos megismerkedésem után elkészítettem néhány alkalmazást, amelyekben megpróbáltam a rendelkezésre álló production node-ok minél szélesebb skáláját használni. Az alábbiakban vázlatosan ismertetem az elkészült programok felépítését. Mindegyik alkalmazás egy közös makefile-t használ, amely hozzáadja az összes dependenciát a fordításhoz, így mindegyik alkalmazás a saját főkönyvtárából kiadott „make” paranccsal fordítható, majd a bin könyvtár alatt lévő binárist indítva futtatható.

5.1. MiddlePoint

A MiddlePoint alkalmazás egy DepthGenerator típusú production node-ot inicializál, és ennek a production node-nak használja az Error state of sensor capability-jét. Az alkalmazás indulásakor felépít egy production chaint, majd a DepthGenerator production node-ot az adatok küldésére utasítja. Ezeket az adatokat egy DepthMetaData típusú objektumba menti (16 bites mélységi adatok). Ezután az alkalmazás ebből az adathalmazból kiválasztja a középső pontot (azaz a szenzor által látott tartomány közepét), majd ennek alapján közli a felhasználóval, hogy milyen messze tartózkodik a szenzortól. Ha ez a távolság kisebb, mint 1 méter, akkor a felhasználó túl közel van a szenzorhoz, ha ez a távolság 1 és 5 méter között van, akkor a távolság ideális, ha 5 méternél nagyobb (vagy nincs senki a szenzor előtt), akkor pedig túl távol van a szenzortól. Ezen kívül az alkalmazás közli a felhasználóval, hogy hányadik képkocka adatait látja éppen, és az éppen aktuális FPS (frames per second) értéket is kiírja a képernyőre.

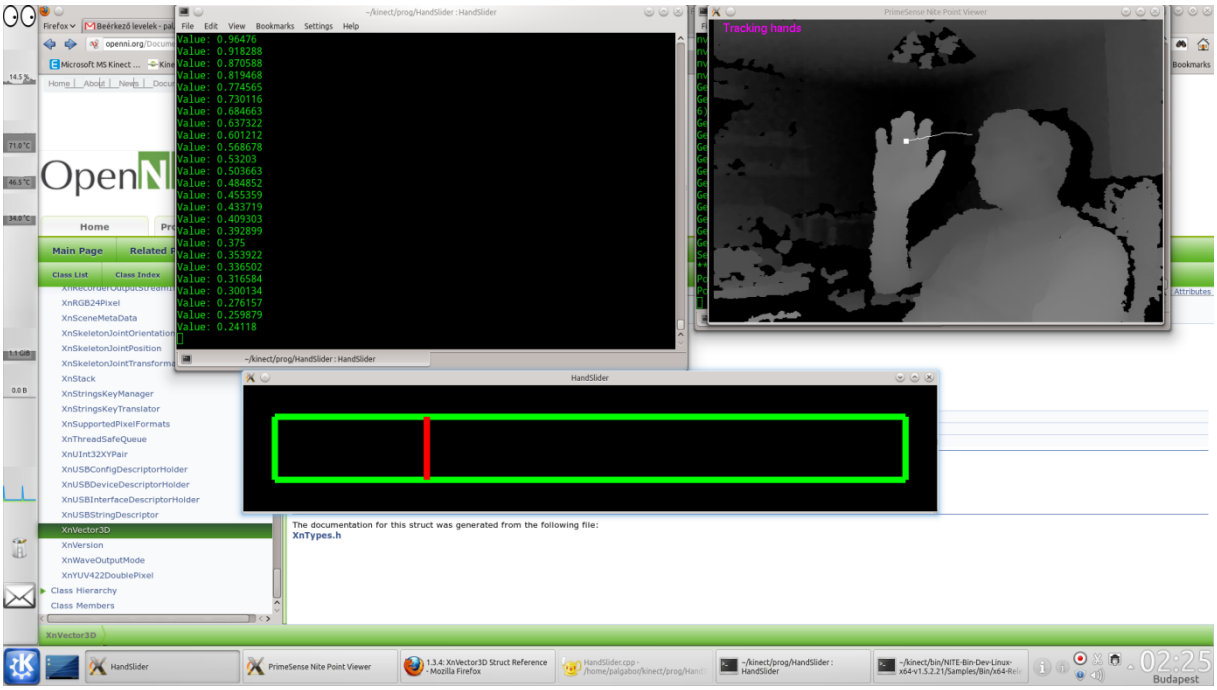


10. ábra: A MiddlePoint alkalmazás futás közben

5.2. HandSlider

A HandSlider nevű alkalmazás a MiddlePoint-tal ellentétben az inicializáláshoz egy Config.xml nevű fájlt használ, az ebben található információk alapján építi fel az általa használt production chaint. Az alkalmazás két production node-dal kommunikál: egy DepthGeneratorral és egy HandTrackerrel. A DepthGenerator production node mirror nevű capability-jét is felhasználja, a bejövő adatok tükrözéséhez.

Mivel a HandTracker Analizátor típusú production node, ezért nem küld folyamatosan adatokat az alkalmazás számára, hanem különböző callback függvényeket kell implementálni, amelyek egy-egy esemény bekövetkeztekor meghívódnak. Az alkalmazás a GLUT (OpenGL Utility Toolkit) segítségével egy zöld téglalapban mozgó piros sávot rajzol ki a képernyőre, amelyet horizontális irányban mozgathatunk a kezünk segítségével. Az alkalmazás közben a háttérben futó konzolra egy 0 és 1 közötti 6 tizedes jegy pontosságú számot ír ki. Itt 0 a legbaloldalibb, 1 pedig a legjobboldalibb érzékelhető pozíciónak felel meg, köztük pedig lineárisan változik a szám értéke. Az alábbi képen egy példaprogram felhasználásával illusztrálom, hogy milyen képet lát pontosan a Kinect az alkalmazás futása közben.



11. ábra: A HandSlider alkalmazás futás közben

6. Egyéb felhasználási területek

Habár a Kinect egyre népszerűbbé válik alkalmazásfejlesztési (nem játékfejlesztési) feladatok terén is, egyelőre kevés gyakorlati alkalmazási területről lehet beszámolni. Szerencsére ezek köre egyre inkább nő. Az alábbiakban felsorolok néhány területet, ahol már jelenleg is sikerrel alkalmazzák a Kinectet.

- Vakok tájékozódásának segítése: http://www.hwsw.hu/hirek/46347/microsoft-kinect-xbox-360-tudomany.html
- Human activity detection: http://pr.cs.cornell.edu/humanactivities/
- SmartTripod: Az MS Robotics Studio segítségével készített, 4 szabadsági fokkal rendelkező robot, amely egy operatőrt képes helyettesíteni, amint az egy mozgó embert filmez: http://prog.hu/hirek/3004/A+kameramant+kivaltani+kepes+robot+nyerte+a+Microsoft+robotikai+fejlesztoversenyet.html

7. Felhasznált irodalom

Andrew Davison – Kinect Open Sourse Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java

Jeff Kramer, Nicolas Burrus, Daniel Herrera C., Florian Echtler, Matt Parker – Hacking the Kinect

http://openni.org/Documentation/ProgrammerGuide.html

http://www.primesense.com/nite

http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx

http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/31/kinect-for-windows-is-now-available.aspx

http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/31/kinect-for-windows-is-now-available.aspx

http://msdn.microsoft.com/en-us/library/jj131033.aspx

http://msdn.microsoft.com/en-us/library/hh855347.aspx

http://openkinect.org/wiki/Main_Page

http://codelaboratories.com/nui

http://pcforum.hu/hirek/12381/Feltortek+a+Microsoft+Kinectet.html

http://www.hwsw.hu/hirek/46889/microsoft-kinect-sdk-fejlesztes.html

http://prohardver.hu/hir/kinect_for_windows_vegre_megjelent_microsoft.html

Ábrajegyzék: