Anssi Kuoppa

ÄLYPUHELINTEN SENSORIT

Ohjelmointia kiihtyvyys- ja liikkeentunnistussensorilla

Opinnäytetyö
Tietojenkäsittely

Toukokuu 2013

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 31.5.2013
Tekijä(t) Anssi Kuoppa	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittely

Nimeke

Älypuhelinten sensorit: Ohjelmointia kiihtyvyys- ja liikkeentunnistussensorilla

Tiivistelmä

Tämän opinnäytetyön tarkoituksena on ottaa selvää, mitkä ovat tärkeimmät tämän hetkiset älypuhelinten sensorit sekä mihin ja miten niitä käytetään hyödyksi. Pääpaino tutkimuksessa on kuitenkin kiihtyvyyssensorille ja liikkeentunnistussensorille ohjelmoinnissa. Työni teoriaosuudessa käyn läpi pääpiirteittäin tärkeimmät älypuhelinten sensorit ja keskityn käsittelemään tarkemmin kiihtyvyyssensorin ja liikkeentunnistussensorin ja varsinkin niiden ohjelmointiaspektin.

Käytännön esimerkkinä tein minua eniten kiinnostavan sensorin eli kiihtyvyyssensorin pohjalta pelidemon. Käyn läpi, miten XNA:lla voidaan rakentaa Windows Phone 7.1 kehitysympäristöä käyttäen kiihtyvyyssensoriin pohjautuva peli. Tämän lisäksi tutustun valmiin esimerkin avulla, miten liikkeentunnistussensori otetaan haltuun Microsoft Visual Studiossa. Päätännössä käsittelen erilaisin esimerkein älypuhelinten ja sensoreiden nykyistä tilaa ja tulevaisuuden näkymiä sekä pureudun tekemäni käytännön esimerkin kehitysmahdollisuuksiin.

Asiasanat (avainsanat)

älypuhelimet, Windows Phone, kiihtyvyys, liikkeentunnistus, Visual Studio

Sivumäärä	**Kieli**	**URN**
31	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Jukka Selin	**Opinnäytetyön toimeksiantaja** Mikkelin ammattikorkeakoulu

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** 31th of the May 2013
Author(s) Anssi Kuoppa	**Degree programme and option** Business Information Technology

Name of the bachelor's thesis

Smartphones' sensors: Programming with accelerometer and motion detection sensor

Abstract

The purpose of this bachelor's thesis was to find out which were the most important sensors of smartphones and how and where to use them. The main focus was on programming with an accelerometer and a motion detection sensor. The theoretical part of my work went through the general outline of the most important sensors of smartphones. The theoretical part concentrated on the accelerometer and the motion detection sensor and especially the aspect of the programming.

As the practical part of my thesis I made a game demo with the accelerometer, which was the most interesting sensor to me. I presented how to build an accelerometer game with XNA by using the Windows Phone 7.1 development environment. The practical part also showed with a ready example how to use the motion detection sensor in Microsoft Visual Studio. The end of the thesis discussed the present state of sensors and their future. I also surveyed the development possibilities of my game demo.

Subject headings, (keywords)

smartphones, Windows Phone, acceleration, motion detection, Visual Studio

Pages	Language	URN
31	Finnish	

Remarks, notes on appendices

Tutor	Bachelor's thesis assigned by
Jukka Selin	Mikkeli University of Applied Sciences

SISÄLTÖ

1 JOHDANTO

Opinnäyteyössäni perehdyn tärkeimpiin tämän hetkisiin älypuhelinten sensoreihin ja niiden toimintaperiaatteisiin painottaen ohjelmointia, keskittyen kuitenkin tarkemmin vain kiihtyvyyssensoriin ja liikkeentunnistussensoriin. Älypuhelimet ovat kehittyneet rajusti viime vuosina, minkä takia aihe on hyvinkin uusi ja tärkeä. Kiinnostuin sensoreista älypuhelimissa omien älypuhelinkokemusten kautta. Tämän takia tekstini sopii hyvin henkilöille, jotka ovat kiinnostuneita tietämään hieman enemmän älypuhelimista ja niiden sensoreista ja varsinkin kiihtyvyyssensorista ja liikkeentunnistussensorista. Itselläni on tällä hetkellä Nokia Lumia 800 Windows Phone -käyttöjärjestelmällä, jolle myös kehitän esimerkkidemoni tässä työssä.

Tutkimusongelmanani on selvittää, mitkä ovat älypuhelinten tärkeimmät sensorit tällä hetkellä sekä mihin ja miten niitä käytetään hyödyksi älypuhelimissa ja ohjelmissa. Erityisesti keskityn tutkimusongelmassani kahteen sensoriin, jotka ovat kiihtyvyyssensori ja liikkeentunnistussensori, ja eritoten näiden hyödyntämiseen ohjelmoinnissa. Opinnäytetyöni toimeksiantajana toimi Mikkelin ammattikorkeakoulu.

Työni alkaa teoriaosuudella, jonka ensimmäisessä osiossa käyn läpi älypuhelinten sensorit yleisesti, minkä jälkeen kerron jokaisesta tärkeämmästä sensorista hieman tarkemmin. Jokaisesta sensorista esittelen esimerkin, missä ja miten kutakin sensoria voidaan käyttää älypuhelimessa hyödyksi. Seuraavassa osiossa eli luvussa kolme otan ensimmäiseksi alkukosketuksen kolmen tämän hetken kuumimman älypuhelinalustan eli Windows Phone:n, Androidin ja iOS:n eroavaisuuksiin ja yhtäläisyyksiin. Tämän jälkeen käsittelen tarkemmin kiihtyvyyssensorin ja liikkeentunnistussensorin pureutumalla niiden toimintaperiaatteisiin rautatasolla ja koodillisesti sekä niiden tuomiin mahdollisuuksiin esimerkein.

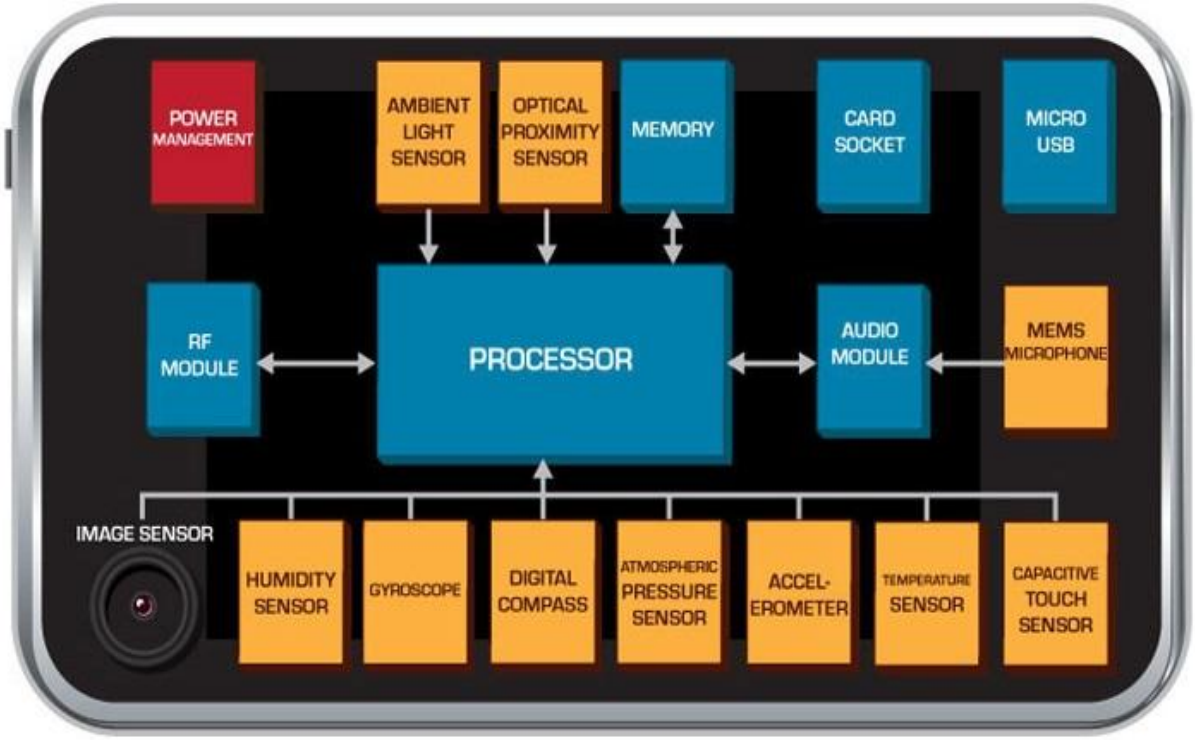
Lopuksi kolmannessa luvussa paneudun aiemmin mainittujen kolmen alustan koodilliseen antiin tarkemmin eli miten kullakin alustalla otetaan sensorit käyttöön sovellukseen, varsinkin kiihtyvyyssensori ja liikkeentunnistussensori. Työni neljäs luku käsittelee aluksi käytännön toteutuksena tekemääni kiihtyvyyssensoriin pohjautuvaa Windows Phone -demoa. Tässä kyseisessä luvussa käyn vaihe vaiheelta läpi miten XNA:lla pystytään tekemään 3D-peli Windows-puhelimelle, mikä käyttää hyödyksi kiihtyvyyssensoria. Tämän jälkeen neljännessä luvussa käsittelen vielä Windows Pho-

ne kehittäjäympäristön mukana tulevan emulaattorin tuomia etuja kiihtyvyyssensoria testatessa ja miltä demo näytti emulaattorin ruudulla eri vaiheissa, ja mitä ruuduilla tapahtuu. Neljännen luvun lopussa käyn valmiin koodin pohjalta läpi vaiheittain, miten liikkeentunnistussensoriin pohjautuvan sovelluksen luontia kannattaa lähestyä Visual Studiossa. Viimeisessä luvussa eli päätännössä annan tutkimusongelmaani ratkaisun ja käsittelen hyvinkin laajasti älypuhelinten ja eri sensoreiden tulevaisuuden näkymiä omasta mielestäni sekä erilaisten esimerkkien avulla. Tämän lisäksi pohdin tekemäni kiihtyvyyssensoridemon kehitysmahdollisuuksia.

2 YLEISESTI ÄLYPUHELINTEN SENSOREISTA

Tässä luvussa käyn läpi älypuhelinten yleisimmät sensorit tällä hetkellä. Tiedostan kuitenkin, että älypuhelimessa on enemmänkin sensoreita ja niiden määrä kasvaa jatkuvasti. Älypuhelinten sensorit perustuvat teknillisiin komponentteihin, jotka reagoivat tietyllä tavalla. Rautatasolla sensorit ovat hyvin pitkälti samanlaisia ja käyttäytyvät samalla tavalla. (Zhou ym. 2011, 237–238.) Mukaan lukien muissa laitteissa olevat sensorit, jotka käyttävät samanlaiseen teknologiaan perustuvia sensoreita kuten autot (Mechanical Engineering 2013). Yleensä älypuhelinten sensorit on pakattu mikrosysteemikomponentteihin, jotka saattavat tunnistaa montakin eri asiaa (Farnell 2013). Eri käyttöjärjestelmiin tuotavat pienoiset erot tapahtuvat siis käyttöjärjestelmän puolella eli eri alustoilla. Sensorit ovat kuitenkin laitekohtaisia, minkä takia sensoreiden määrä saattaa vaihdella puhelinmalleittain. Myös mahdollisuus sensoreiden käyttöön sovelluskehityksessä saattaa vaihdella käyttöjärjestelmittäin. (Zhou ym. 2011, 237–238, 245.)

Yleisimmät ja ohjelmoitavissa olevat sensorit älypuhelimissa tällä hetkellä voi nähdä lähes kaikki kuvasta 1, mutta käyn ne nyt läpi päällisin puolin. GPS on sensori, joka perustuu langattomaan verkkoon ja satelliitteihin. Kiihtyvyyssensori on sensori, joka mittaa nimestä poiketen puhelimen asentoa. Liikkeentunnistussensori eli gyroskooppi mittaa puhelimen kiihtyvyyttä. (Jaman 2011.) Näiden lisäksi älypuhelin pitää sisällään digitaalisen kompassin, mikrofonin ja monikosketuksentunnistussensorin sekä erilaiset kuvasensorit, joihin kamera puhelimissa perustuu. Kaksi vähemmän tunnettua sensoria, mutta paljon käytettyä, ovat valaistuksentunnistussensori ja etäisyyssensori. (Farnell 2013.)



KUVA 1. Älypuhelimen komponentit ja niiden sijoittelu (Farnell 2013)

Kuvassa 1 esitellään yleinen sijoittelutapa monille eri älypuhelinten sensoreille, josta käy ilmi, että paras sijoituspaikka sensorille ei ole keskellä puhelinta vaan reunemmalla. Siten, että herkkä sensori ei altistu liialle kuumuudelle tai iskuille. Käyn seuraavaksi läpi kaikki aikaisemmin tässä luvussa luetellut sensorit hieman tarkemmin. (Farnell 2013.)

2.1 Kiihtyvyyssensori

Kiihtyvyyssensorilla (englanniksi accelerometer) voidaan mitata puhelimen asentoa ja kallistuksia. Sensori laskee liikkeet suhteessa maapallon pintaan painovoiman avulla ja ilmoittaa puhelimen asennot ja kallistuksen x-, y- ja z-akseleiden mukaan. Kiihtyvyyssensori on nykyisin vakiovaruste älypuhelimessa, ja sitä hyödynnetään esimerkiksi selaimissa ja tekstiviestiä syötettäessä. Puhelinta käännettäessä tiettyyn asentoon selain, tekstinsyöttökenttä ja näppäimistö kääntyvät automaattisesti mukana. Myös älypuhelimen kamera käyttää hyödyksi kiihtyvyyssensoria kuvaa otettaessa, minkä avulla kamera tietää, onko puhelin pysty- vai vaaka-asennossa. (Jaman 2011.)

KUVA 2. iBeer-sovellus (iTunes 2013)

Koska kiihtyvyyssensori on ollut jo pidemmän aikaan käytössä älypuhelimissa, niin siihen perustuvia ohjelmiakin on tehty hyvin paljon, varsinkin pelejä. Esimerkiksi Pocket God -niminen peli, jossa voi kallistamalla puhelinta eri suuntiin, vierittää saaren asukkaita hukkumaan ja erilaisiin loukkuihin sekä ravistamalla aiheutetaan maan järistys. (Hornshaw 2010.) Omanlaistansa hupia tarjoavat myös erilaiset ”juomasimulaattorit”, joissa puhelinta kallistamalla juoman voi juoda kuin oikeasti. Esimerkiksi iBeer, minkä toimintaperiaatteen näkee kuvasta 2. (iTunes 2013.)

2.2 Liikkeentunnistussensori

Älypuhelinten liikkeentunnistussensori eli gyroskooppi (englanniksi motion detection sensor eli gyroscope) on elektroninen sensori, joka mittaa puhelimen liikettä ja kiihtyvyyttä. Sensorin toiminta perustuu gyroskoopin toimintaperiaatteeseen. Tämä tarkoittaa sitä, että gyroskoopilla voidaan havaita sellaiset liikkeet, mitä ei kiihtyvyyssensorilla pysty eli puhelimen pyöritysliikkeet ja paikan vaihtelut. Gyroskoopin avulla pystytään esimerkiksi havaitsemaan kätevästi puhelimen putoamisen. Hyvin usein gyro-

skooppia käytetään kiihtyvyyssensorin kanssa yhdessä, jolloin saadaan tarkemmat tiedot puhelimen erilaisista liikkeistä sovellukselle. (Jaman 2011.)

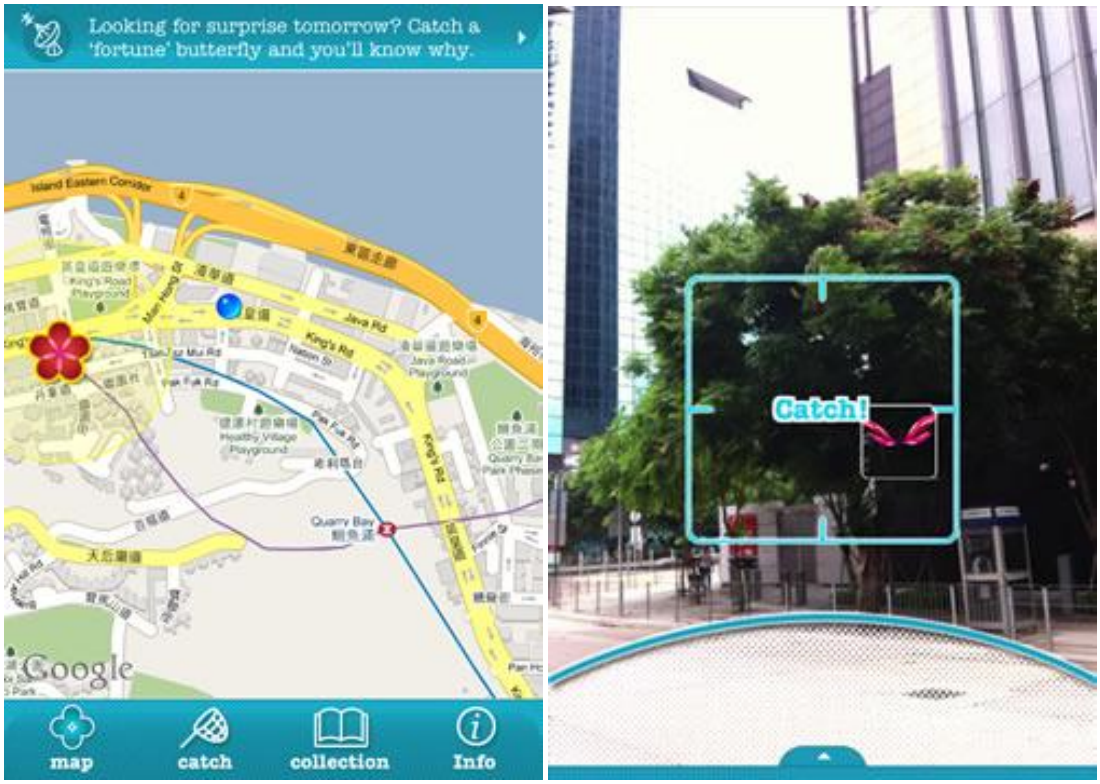
KUVA 3. Ultimate Mouse tominnassa (Negusoft 2013)

Vaikka liikkeentunnistussensori on vasta muutama vuosi sitten yleistynyt älypuhelimissa, on sitä hyödyntäviä ohjelmia jo moneen lähtöön. Pelirintamalla gyroskooppia on hyödyntänyt muun muassa suosittu ja näyttävä ensimmäisenpersoonan ammuntapeli N.O.V.A. Kyseisessä pelissä gyroskooppi tuo tarkkuutta tähtäykseen ja uudenlaisen tavan ohjata hahmoa. (iTunes 2013.) Hyvinkin edistyksellinen Ultimate Mouse -ohjelma hyödyntää gyroskoopin ominaisuuksia niin, että puhelinta voi käyttää tietokoneen hiirenä samalla tavalla kuin liikkeentunnistusohjaimia konsolissa kuten esimerkiksi Wii Remote. Kuvassa 3 on Ultimate Mouse toiminnassa. (Negusoft 2013.)

2.3 GPS

Älypuhelinten GPS (Global Positioning System) perustuu puhelimen sisällä olevaan niin sanottuun paikannuspiiriin (Järvinen 2012, 152). Paikannuspiirissä olevan paikannussensorin avulla pystytään paikantamaan satelliitit ja havaitsemaan puhelintornit sekä langattomat tukiasemat. Yleensä GPS:n toimintaan tarvitaan vain satelliittipaikannus, jolloin Internet-yhteyttä ei tarvita, mutta jos nykyinen sijainti halutaan selvittää sisätiloissa ja katvealueella, voidaan Internet-yhteyden ollessa auki puhelin paikantaa puhelinmastojen ja langattomien tukiasemien avulla. (Schirmer & Höpfner 2013,

7-9.) Satelliitilta saadun signaalin avulla GPS – vastaanotinsensori puhelimessa laskee puhelimen sijainnin kolmiomittauksella (Jaman 2011).



KUVA 4. iButterflyn kartta ja lisättytodellisuus (iButterfly Honkong 2013)

GPS:ää käytetään siis paikantamaan puhelin. Puhelimen paikannusta voidaan taas käyttää hyväkseen esimerkiksi erilaisissa karttasovelluksissa. Hyvin yleisiä ovat esimerkiksi erilaiset kuntoiluohjelmat kuten suosittu Sports Tracker, joka pitää yllä tietoa kuljetuista reiteistä GPS:n avulla (Sports Tracker 2013). Ohjelmapuolella on hyvin paljon erilaisia GPS:n perustuvia sovelluksia, mutta myös pelejä löytyy. Hyvänä esimerkkinä GPS–pelistä on iButterfly, jossa tarkoituksena on oikean kartan avulla etsiä lähellä olevia virtuaaliperhosia ja napata ne lisätyntodellisuuden avulla, eri alueilla on myös omat perhoslajinsa (iButterfly Honkong 2013). Kuvassa 4 näkyy iButterfly:n kartta- ja kameranäkymä. GPS–sensoria voidaan käyttää nykyään hyödyksi myös erilaisissa tutkimuksissa. Kanadassa esimerkiksi älypuhelimia laitettiin putkiloihin, joissa oli akku ja paino sekä GPS–sovellus päällä, ja heiteltiin jokeen. Näin saatiin hyvinkin tarkkaa tietoa veden virtauksista ja muusta joen käyttäytymisestä. (Yang 2012.) USA:n armeija on taas esitellyt tänä vuonna GPS–sirun, joka on pienempi kuin pennin kolikko ja ei tarvitse laisinkaan satelliitteja avukseen paikantaessaan sijaintia (Prigg 2013). Kenties tässä on myös älypuhelinten tulevaisuus?

2.4 Kuvasensori

Kameraa itsessään ei voida oikeastaan nimittää älypuhelimissa sensoriksi, vaan niissä olevia kuvasensoreita (englanniksi image sensor), joita on paljon erilaisia ja ne sisältävät paljon erilaisia ominaisuuksia. Kuvasensorin yhteydessä ovat myös linssi ja kameran mikrosysteemi. Sensorien ominaisuudet vaihtelevat valmistajittain ja samoja sensoreita käytetään yleensä myös digitaalikameroissa ja tablettitietokoneissa. (Chitkara & Chin 2013.) Kuvasensorin avulla voidaan esimerkiksi parantaa valaistusta ja kuvanlaatua, kuten kontrastia kuvaa ottaessa ja uusimmissa myös kuvan ottamisen jälkeen, esimerkkinä muun muassa Sonyn uudet CMOS sensorit (Sony 2012).

Toshiban tulevassa sensorissa on esimerkiksi ominaisuus, joka mahdollistaa kuvan tarkentamisen jälkikäteen, käyttäjän valitsemasta kohdasta (Owano 2012). Kuvasensorit mahdollistavat kuvien ainutlaatuisen muokkauksen, laadun ja ominaisuuksien kehittymisen (Chitkara & Chin 2013). Sain kuitenkin huomata, että suoranaisia ohjelmia niihin perustuen ei ole nimeksikään. Erilaisia innovaatioita kuitenkin löytyy paljonkin, esimerkiksi uuden Samsung Galaxy S4:n etukameran sensori pystyy havaitsemaan silmien liikkeet (Shah 2013).

2.5 Valaistuksentunnistus- ja etäisyyssensori

Valaistuksentunnistussensori (englanniksi ambient light sensor) tunnistaa ympäristöstä tulevan valon määrän ja säätelee siten älypuhelimen näytön kirkkautta akkuystävällisemmäksi. Useimmissa älypuhelimissa valoon reagointi tapahtuu usean valodiodin avulla, mitkä reagoivat kukin vain tiettyyn valon spektriin. Näitä diodeja yhdistelemällä matemaattisesti sensori pystyy antamaan hyvinkin tarkkoja tuloksia laitteelle, jonka perusteella voidaan toimia. Älypuhelimessa valaistuksentunnistussensorin tarkoitus on lähinnä säästää puhelimen akkua. (Jaman 2011.)

Valaistuksentunnistussensori on tärkeässä roolissa älypuhelimen akun kestävyyden kannalta, mutta sovellusten kirjolla ei voida tämän sensorin kohdalla juhlia, sillä suurin osa sovelluksista vain mittaa ja ilmoittaa ympäröivän valon määrän luxeina, kuten esimerkiksi Androidille saatavilla oleva Light Meter. (Google Play 2013) Valaistuksentunnistussensoria kuitenkin käytetään myös muissa elektronisissa laitteissa, esi-

merkiksi autoissa, joissa se säätää automaattisesti muun muassa auton etuvaloja (Mechanical Engineering 2013).

Etäisyyssensori (Proxmity sensor) on hyvin hyödyllinen sensori, jota pääsääntöisesti käytetään älypuhelimissa tunnistamaan kuinka lähellä puhelin on jotakin objektia esimerkiksi ihmistä. Tämän ansiosta puhelinta esimerkiksi korvalle vietäessä pystytään automaattisesti lukitsemaan näyttö ajoissa, eikä puhelimella tehtyjä virhepainalluksia tule ja akkua säästyy. (Jaman 2011.) Etäisyyssensoriin perustuvat ohjelmat toimivat kaikki samalla periaatteella eli etäisyyssensori pitää peittää jollain, mikä sitten aiheuttaa jotain. Esimerkiksi PROXIMITY Screen Off sovelluksen avulla näyttö lukkiutuu automaattisesti, kun sensorin eteen vie jotain eli esimerkiksi taskuun laitettaessa näyttö menee lukkoon ja pois otettaessa se on käyttövalmiina. Proximity Plus Plus -nimisellä sovelluksella voi sen sijaan soittaa, lopettaa ja vaihtaa kappaletta puhelimen musiikkisoittimessa lennossa vain pyyhkäisemällä kädellä etäisyyssensorin yli. (Google Play 2013.)

2.6 Monikosketuksentunnistussensori

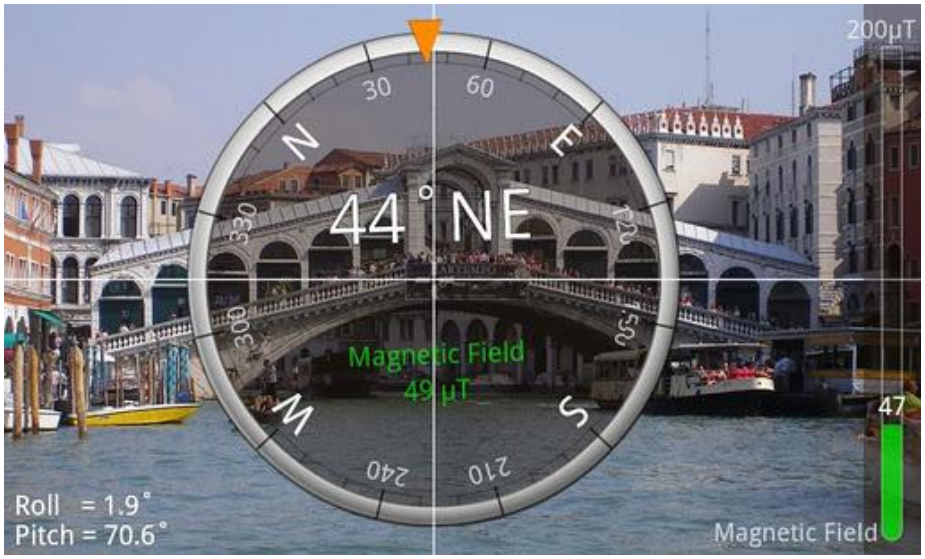
Monikosketuksentunnistussensorit (englanniksi multi-touch sensor) mahdollistavat älypuhelinten kosketusnäyttöjen käytön useammalla kuin yhdellä sormella. Monikosketus perustuu sensoreihin, jotka pystyvät reagoimaan havaitessaan sähköstaattisen muutoksen tietyssä pisteessä, jonka esimerkiksi sormella koskettaminen aiheuttaa. (Larsen 2012.) Sensorit saattavat myös toimia tunnistamalla läheisyyden, kuten läheisyyssensori tai tunnistamalla painalluksen (Atmel 2010). Sensorit sijaitsevat yleensä erillisellä kosketuspaneelilla näytön lasin ja lcd-paneelin välissä. Kuitenkin muutamissa uusimmissa puhelimissa, kuten iPhone 5, sensorit on jo saatu lisättyä itse lcd-paneeliin, mikä mahdollistaa ohuemmat puhelimet. (Larsen 2012.)

KUVA 5. MultiTouchin valmistama monikosketusnäyttöseinä (MultiTouch 2013)

Monikosketuksen ansiosta, esimerkiksi SmartMouse -nimisen ohjelman käyttäminen on mahdollista. Ohjelmalla voidaan käyttää puhelinta hiirenä, hyödyntäen monikosketuksen tuomia mahdollisuuksia kuten kahden sormen painallusta. (Dong 2013.) Monikosketukseen soveltuvia sensoreita hyödynnetään myös isommissa näytöissä. Suomalainen MultiTouch OY on monikosketuksen moniosaaja ja valmistaa suuriakin monikosketuksentunnistukseen perustuvia näyttöjä, hyvänä esimerkkinä maailman suurin monikosketusnäytön valmistaminen. Kuvassa 5 on MultiToucin tällä hetkellä suurin valmistama julkisella paikalla oleva monikosketusseinä. (Good News from Finland 2012.)

2.7 Digitaalinen kompassi ja mikrofoni

Digitaalinen kompassi (englanniksi digital compass) älypuhelimissa ei perustu tavallisiin magneetteihin kuten kompassit yleensä, koska magneetit sotkisivat älypuhelimen toiminnan. Sen sijaan kompassi perustuu sensoreihin, jotka toimintatavasta riippuen keräävät tietoa siitä, missä suunnassa magneettinen pohjoinen on. Esimerkiksi Applen uusissa älypuhelimissa sensori etsii tiettyä alhaista taajuutta, joka on tyypillinen pohjoisnavalle. Kompassi kuitenkin tarvitsee poikkeuksetta avukseen kiihtyvyyssensorin, jotta se pystyy laskemaan oikean suunnan. Esimerkiksi iPhone 4 tunnistaa suunnan suhteessa maan magneettikenttään jännitteen muutosten avulla. (Jaman 2011.) Kompassia harvemmin käytetään muunlaisissa ohjelmissa, kuin erilaisissa kompassisovelluksissa. Esimerkiksi Smart Compass Androidille, mikä yhdistää kätevästi kompassin kameranäkymään, kuva 6 havainnollistaa tätä. (Google Play 2013.)

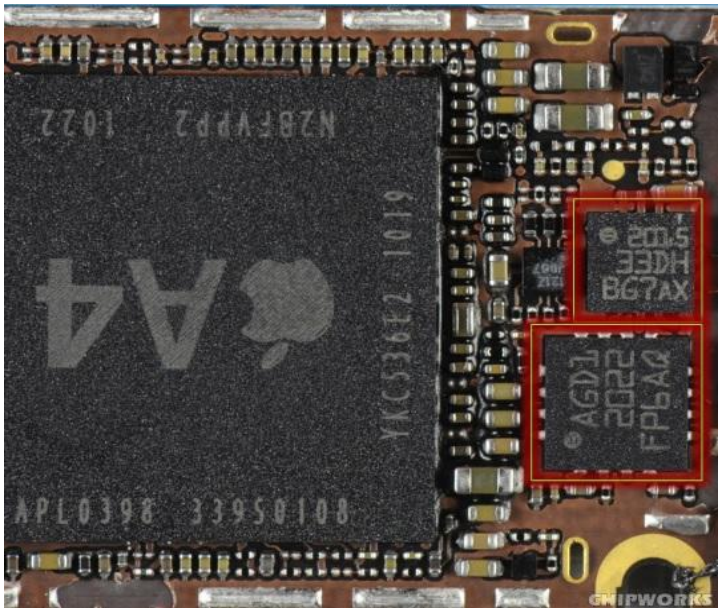


KUVA 6. Smart Compass toiminnassa kameranäkymässä (Google Play 2013)

Normaalia mikrofonia (englanniksi microphone) ei mielestäni voida pitää sellaisenaan sensorina. Älypuhelinten mikrofonit kuitenkin tekevät paljon muutakin nykyään kuin välittävät äänen eteenpäin. Mikrofonit perustuvat sensoreihin, jotka pystyvät välittämään ääneen suodatettuna ja minimoimaan taustahälyn. Tämän lisäksi sensorit pystyvät tunnistamaan ihmisen ääntä tietyissä rajoissa. (OKI 2013.) Esimerkkeinä muun muassa tähän perustuvat ohjelmat, kuten iPhonen Siri ja Googlen puheentunnistuksella toimiva haku. Älypuhelimen mikrofonia voidaan käyttää myös hyödyksi muissa laitteissa, esimerkiksi WO Mic -niminen ohjelma mahdollistaa älypuhelimen mikrofonin käytön tietokoneen mikrofonina (Technomiz 2013).

3 KIIHTYVYYS- JA LIIKEENTUNNISTUSSENSORI ERI ALUSTOILLA

Tässä luvussa käsittelen tarkemmin mielestäni kaksi mielenkiintoisinta sensoria eli kiihtyvyys- ja liikkeentunnistussensorin. Erityisesti kiihtyvyyssensoria, johon olen demossani keskittynyt työni loppupuolella. Tällä hetkellä eniten näkyvillä olevat älypuhelin alustat eli käyttöjärjestelmät ovat Android, Windows Phone ja iOS. Sen takia käsittelenkin lähemmin vain kyseisiä alustoja kiihtyvyys- ja liikkeentunnistussensorin osalta, tiedostaen kuitenkin, että markkinoilla on muitakin suhteellisen suosittuja käyttöjärjestelmiä, kuten Symbian, Bada ja BlackBerry. (Ahonen 2013.) Rautatasolla Applen iOS puhelimet eroavat kahdesta kilpailijastaan, koska Apple kontrolloi puhelimen valmistusta. Näin ollen myös kiihtyvyys- ja liikkeentunnistussensorit ovat aina Applen valitsemia, eli karrikoidusti aina saa samat sensorit.



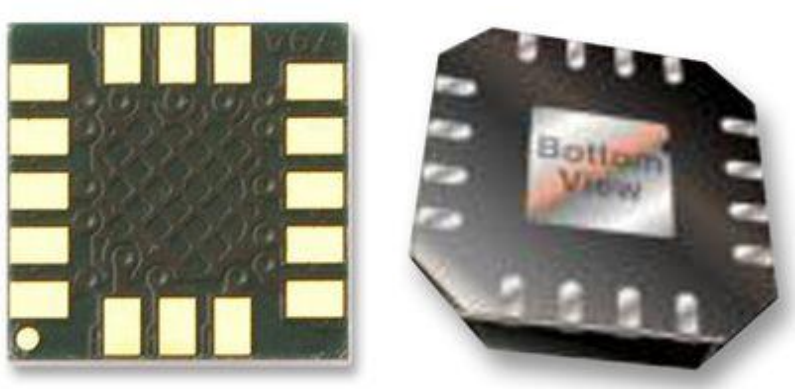
KUVA 7. Kiihtyvyyssensori ja gyroskooppi iPhone 4:ssä (ChipWorks 2010)

Android - ja Windows -puhelinten alkuperäiset laitevalmistajat sen sijaan saavat valita käyttämänsä raudan vapaammin, jolloin rauta tasolla esimerkiksi Android -puhelinten kesken saattaa olla isompiakin eroja, vaikka käyttöjärjestelmä olisi täysin sama. Yleensä kuitenkin raudassa käytetty tekniikka on nykyään hyvin samanlaista, sensoreista riippumatta. (Zhou ym. 2011, 237.) Kuvassa 7 näkyy rautatasolla punaisella neliöitynä iPhone 4:n kiihtyvyyssensori ylhäällä ja liikkeentunnistussensori alhaalla.

3.1 Kiihtyvyyssensorin salat

Kiihtyvyyssensori on älypuhelimen yksi merkittävimmistä sensoreista, ja siksi myös suurimman kiinnostukseni kohde. Kyseistä sensoria käytetään, jos halutaan tietää missä asennossa puhelin on. Sensori siis reagoi puhelimen kääntelyihin, mutta myös ravistukseen. Kiihtyvyyssensoria käytetään hyödyksi puhelimen aivan perusasioissa, kuten virtuaalinäppäimistön kääntämisessä, kuvien ja kameran kääntämisessä oikeaan asentoon, pystyyn tai vakaan. Ominaisuutta käytetään myös paljon hyödyksi peleissä, joissa ohjataan jotain, esimerkiksi autoa ajopeleissä. Myös tekemäni demo perustuu juuri kiihtyvyyssensoriin. (Jaman 2011.)

Kiihtyvyyssensorin toiminta perustuu yleensä, johonkin elementtiin, joka pääsee liikkumaan tietyissä rajoissa. Kyseessä voi esimerkiksi olla piimetallia, joka on kiinnitetty tietyistä pisteistä alustaan, mutta kuitenkin niin, että ne pystyvät liikkumaan havaitun kiihtyvyyden suuntaan. (ST 2009, 15.) Sovellusten kehittäjä pääsee käsiksi kiihtyvyyssensoriin jokaisella kolmella alustalla vaivatta, koska kiihtyvyyssensori on hyvin suuressa roolissa älypuhelimissa jo käyttömukavuuden kannalta. iOS:lla kiihtyvyys sensoriin pääsee käsiksi ”CMMotionManager” luokalla (Lee 2011, 344). Kiihtyvyyssensoriin pääsee käsiksi Windows 7.1 kehittäjäympäristössä käyttämällä kirjastoa ”Microsoft.Devices.Sensors” (Järvinen 2012, 142). Androidilla on kaikista kolmesta helpoin päästä käsiksi mihin tahansa tuettuun sensoriin ”SensorManager” luokan avulla (Zhou ym. 2011, 247).



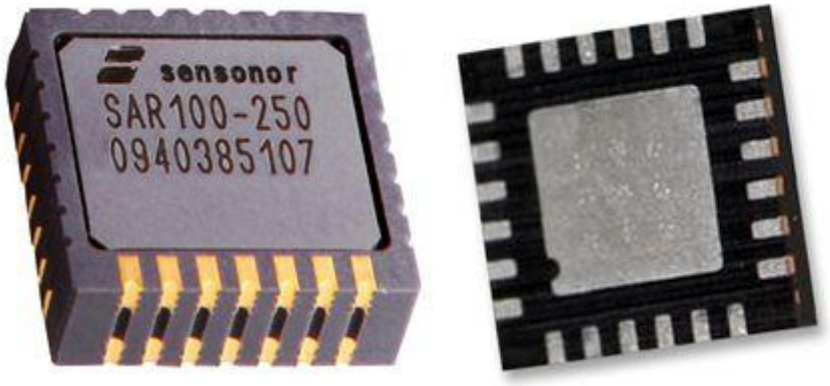
KUVA 8. Kahden eri valmistajan kiihtyvyyssensorit (Farnell 2013)

Kiihtyvyyssensoria älypuhelimessa voidaan käyttää moniin eri tarkoituksiin, hyvänä esimerkkinä muun muassa tunnistamaan kolarit ja lähettämään mahdollisesti automaattisesti tieto puhelimesta eteenpäin (Albright ym. 2013). Samanlaista teknologiaa käytetään myös tietyissä uusissa autoissa, mikä on tosin hieman kalliimpaa (BBC News 2012). Kiihtyvyyssensoriin perustuvia ohjelmiakin on älypuhelimille tarjolla pitkä liuta, koska järjestään kaikki uudet älypuhelimet tukevat sitä. Erittäin suosittu se on ollut erilaisten ajopelien ohjaus järjestelmänä, kuten esimerkiksi Need For Speed Undercover pelissä (Burd 2011). Kiihtyvyyssensoria itsessään on myös kehitetty muun muassa lääketieteen käyttöön, esimerkiksi seuraamaan ihmisen äänen värähtelyä kaulalta älypuhelimen välityksellä. Tarkkaan ottaen tietyn tyyppinen herkkään liikkeeseen reagoiva kiihtyvyyssensori asetetaan käyttäjän kaulalle solisluun yläpuolelle, mikä tunnistaa äänihuulten värähtelyn, josta tieto sitten välitetään älypuhelimeen. (Mehta ym. 2012.) Kahden eri valmistajan kiihtyvyyssensorit voi nähdä kuvasta 8.

3.1 Liikkeentunnistussensorin ihmeet

Gyroskooppi eli liikkeentunnistussensori on suhteellisen uusi tuttavuus älypuhelin markkinoilla. Eikä sitä ole läheskään jokaisessa älypuhelin mallissa. Kuitenkin gyroskooppia käytetään nykyään paljolti sovelluksia tehdessä kiihtyvyys sensorin kanssa yhdessä. Näin saadaan puhelimen liikkeet lähes täydellisesti rekisteröityä. Gyroskooppi pystyy havaitsemaan puhelimen pyöritykset ja mahdollisen kiihtyvyyden, mitä ei kiihtyvyys sensorilla voi mitata. Esimerkiksi jos puhelinta pitää tasaisesti liikkumattomana, kiihtyvyyssensori ei tajuaisi vaikka puhelin liikkuisi johonkin suuntaan tasaista nopeutta (Conway & Hillegass 2010, 119). Ulkonäöllisesi puhelimen gyroskooppi muistuttaa paljolti kiihtyvyys sensoria, kuten kuvasta 9 voidaan todeta ver-

taamalla sitä kuvaan 8. Kyseessä voi olla esimerkiksi gyroskooppi, joka sisältää kaksi itsenäistä värähtelyyn reagoivaa mikrosysteemi liikkeentunnistussensoria, joista toinen rekisteröi x- ja toinen y-akselin pyörähdykset. Rekisteröinti perustuu värähtelyyn, jota tapahtuu kun puhelinta kääntelee. (Inven Sense 2010, 6.) Kuvasta 9 voidaan havaita kahden eri gyroskooppi sensorin rakenne ja ulkomuoto.



KUVA 9. Kahden eri valmistajan liikkeentunnistussensorit (Farnell 2013)

Gyroskooppi on mahdollistanut älypuhelinten käytön mitä erilaisimmin tavoin, kuten esimerkiksi muodostamaan pään liikkeitä vastaavan reaaliaikaisen virtuaalitodellisuuden. FOV2Go on yksi esimerkki tällaisesta rauta- ja ohjelmisto paketista, jonka avulla voi luoda virtuaalitodellisuuden älypuhelimille. (Iliff 2012.) Älypuhelimilla on myös jo paljon ohjelmia, jotka hyödyntävät gyroskoopin tuomia uusia ominaisuuksia, pyörityksestä erilaisiin liikkeisiin. Hyvänä esimerkkinä Flip It! Gyro Game, jonka tarkoituksena on vain näyttää hauskalla tavalla mihin kaikkeen älypuhelimesi gyroskooppi pystyykään, äläkä unohda kavereita (iTunes 2013). Mikrosysteemi gyroskooppia sellaisenaan voidaan myös käyttää moneen tarkoitukseen. Sitä voidaan käyttää esimerkiksi mittamaan koripalloilijoiden tarkan ja reaaliaikaisen suorituksen koripallon sisällä suoraan älypuhelimeen tai tietokoneelle. (Inven Sense 2013)

3.3 Windows Phone

Microsoft Visual Studiolla pääsy Windows -puhelimen liikkeentunnistus- ja kiihtyvyyssensoriin tapahtuu lisäämällä ulkopuolinen ”Sensors” -kirjasto koodiin eli kokonaisuudessaan ”Microsoft.Drvices.Sensosrs”. Tämän jälkeen sensoreille luodaan halutut muuttujat ja ne otetaan käyttöön erikseen eli tässä tapauksessa gyroskooppi ja kiihtyvyyssensori. Kuva 10 havainnollistaa tätä toimenpidettä (Järvinen 2012, 141–147.)

```
using Microsoft.Devices.Sensors;

namespace accelgyrosdemo
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        Gyroscope gyroscope;
        Accelerometer accelSensor;
        public Vector3 accelReader = new Vector3();
```

KUVA 10. Windows Phone, Ukopuolinen kirjasto ja alustus

Molemmat sensorit saadaan käynnistettyä ohjelmassa lisäämällä koodiin uusi tapauskäsittelijä molemmille sensoreille, jotka käynnistetään ”Start();” metodilla, kuten kuvasta 11 voidaan huomata. Molemmat sensorit käynnistetään samalla tavalla try-catch lauseella ”try”:n sisällä, jotta voidaan tarvittaessa tarkistaa tukeeko käyttölaite sensoria tai käynnistyikö sensori, tämä ei kuitenkaan ole pakollista. Gyroskoopista ja kiihtyvyyssensorista saatu data voidaan halutessa yhdistää ”Sensors” -kirjastosta löytyvän ”Motion” luokan avulla. (Järvinen 2012, 141–148.)

```
//...

accelSensor = new Accelerometer();
gyroscope = new Gyroscope();

try
{
    gyroscope.Start();
    accelSensor.Start();

    kiihtAktiivi = true;
}
//...
```

KUVA 11. Windows Phone, sensorin käynnistys

Windows 8 tukee sensoreita entistä paremmin, ja käyttää niin sanottua ”sensor fusion”-tekniikkaa. Tämän avulla pystytään ottamaan huomioon entistä paremmin milloin puhelimen liike on tarkoituksellinen käyttäjän tekemä kääntö- tai ohjausliike. Koodaajat voivat käyttää joko tätä suodatettua tietoa tai perinteistä raakaa dataa suoraan sensoreilta. (Roivas 2012).

3.4 Android

Androidille koodatteassa kaikki käytettävissä olevat sensorit saadaan käytettäväksi lisäämällä ulkopuolisen ”hardware” -kirjaston sensoriluokat eli ” Sensor,SensorEvent, SensorEventListener ja SensorManager”. Sensoreiden hallinnassa kuitenkin tarvitaan vain ”SensorManageria”, jonka avulla saadaan käyttöön kulloinkin tarvittavan sensorin metodit, kuten kuvassa 12, esimerkiksi ”TYPE_GYROSCOPE”. Sensorien data kerätään sensorin kuuntelijan avulla”mySensorListener”, kuva 12 havainnollistaa myös tätä. Saatuja arvoja voidaan käyttää tämän jälkeen ”SensorEventListener”:n avulla. (Zhou ym. 201, 246-248.)

```
import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class OrientationMeasurements extends Activity {
    private SensorManager myManager = null;
    TextView tv;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tv = (TextView) findViewById(R.id.attitude);
        // Set Sensor Manager
        myManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        myManager.registerListener(mySensorListener,
                myManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_GAME);
        myManager.registerListener(mySensorListener,
                myManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE    ),
                SensorManager.SENSOR_DELAY_GAME);
    }
```

KUVA 12. Sensoreiden käyttöönotto Androidilla (Steele ja To 2011, 178.)

Mielestäni Androidin lähestymistapa sensoreihin on koodaamisen kannalta helpoin, koska kaikkiin pääsee käsiksi kätevästi yhden luokan avulla. Toisaalta taas Android emulaattori ei tue sensoreita kunnolla, joten koodatun ohjelman joutuu testaamaan aina puhelimessa. Toisin kuten esimerkiksi XNA:lla joka mahdollistaa muun muassa kiihtyvyyssensorin ja GPS:n testaamisen emulaattorissa (Järvinen 2012, 60)

3.5 iOS

IPhoneille kiihtyvyyssensori on ollut aina vakio varusteena, sen takia myös sen käyttäminen on tehty helpoksi, kuten muillakin alustoilla. Gyroskooppi tuli iPhonelle vuonna 2010, iPhone 4:n myötä. (Lee 2011, 339.) Nyt esitetty lähestymistapa sensoreiden käsittelyyn tuli IOS 4.0:n mukana (Apple 2013). Tätä ennen kiihtyvyyssensoriin pääsi käsiksi ”UIAccelometer” luokan avulla (Conway & Hillegass 2010, 117).

```
#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>

@interface GyroscopeViewController : UIViewController

mm = [[CMMotionManager alloc] init];
if (mm.isDeviceMotionAvailable) {
    mm.deviceMotionUpdateInterval = 1.0/60.0;
    [mm startDeviceMotionUpdatesToQueue:[NSOperationQueue currentQueue]
                            withHandler:^(CMDeviceMotion *motion, NSError *error)
    {
        CMAttitude *currentAttitude = motion.attitude;
        float roll = currentAttitude.roll;
        float pitch = currentAttitude.pitch;
        float yaw = currentAttitude.yaw;

        txtRoll.text = [NSString stringWithFormat:@"%f",roll];
        txtPitch.text = [NSString stringWithFormat:@"%f",pitch];
        txtYaw.text = [NSString stringWithFormat:@"%f",yaw];

        CMAcceleration currentAcceleration = motion.userAcceleration;
        txtX.text = [NSString stringWithFormat:@"%f",currentAcceleration.x];
        txtY.text = [NSString stringWithFormat:@"%f",currentAcceleration.y];
        txtZ.text = [NSString stringWithFormat:@"%f",currentAcceleration.z];
    }];
```

KUVA 13. Sensoreiden käyttöönotto iOS:lla (Lee 2011, 341–343)

Kiihtyvyyssensoriin ja gyroskooppiin pääsee käsiksi koodillisesti iOS alustalla lisäämällä koodiin ulkopuolisen ”CoreMotion” -kirjaston. Luomalla ”CMMotionManager” objektin päästään sensoreiden ominaisuuksiin. Tämän jälkeen on hyvä tarkistaa onko sensorit saatavilla, minkä voi tehdä joko yksitellen molemmille sensoreille tai kuten kuvassa 13, käyttämällä ”DeveiceMotion” -ohjelmointirajapintaa, mikä sisältää sekä gyroskoopin, että kiihtyvyyssensorin tiedot. ”DeviceMotion”:ia edelleen käyttäen, voidaan taas sensoridatan päivitysten kerääminen aloittaa ”startDeveiceMotionUpdatesToQueun” avulla, katso kuva 13. Data kerätään muuttujiin, gyroskoopille ”CMAttitude”:n ja kiihtyvyyssensorille ”CMAccelartion”:n avulla. (Lee 2011, 341–345.)

4 KIIHTYVYYSSENSORIN JA GYROSKOOPIN ESIMERKIT

Tässä luvussa käsittelen yleisesti tekemiäni demoja, joihin pureudun koodillisesti myöhemmin. Tein demot Windows -puhelimille käyttäen Windows Phone 7.1 kehitysympäristöä XNA:lla. Lähden rakentamaan kiihtyvyyssensorin demoa siltä pohjalta, että aluksi sovellan kiihtyvyyssensorin antamaa dataa johonkin yksinkertaisempaan. Tämän jälkeen perus rungon päälle kerrytän vähän lihaa ja saan aikaan yksinkertaisen pelidemon. Demon suunnittelun aloitan siltä pohjalta, että aluksi yritän ymmärtää koodia sen verran, että saan aikaiseksi kiihtyvyyssensoria käyttäen yksinkertaisemman objektin liikuttelun puhelimen näytöllä. Tämän jälkeen rakennan perusmekaniikan päälle pienimuotoisen kiihtyvyyssensori pelin.

Tavoitteenani on luoda lopulliseksi työksi kolmiulotteiseen maailmaan sijoittuva peli, jossa esimerkiksi pallo kulkee kokoajan eteenpäin ja tarkoituksena on väistellä esteitä. Väistely/hahmon ohjaaminen tapahtuu kiihtyvyyssensoria käyttäen. Demossa olisi hyvä olla erilaisia pelitiloja eli esimerkiksi jonkinlainen päävalikko, itse peli ja lopetusruutu. Pelillisiä sääntöjä olisi myös hyvä olla, kuten jos hahmo osuu esteisiin tai seiniin pallo palaa alkuun tai peli loppuu eli tulee lopetusruutu. Lopetusruudussa voisi näkyä kerätyt pisteet, kerätyt osumat tai vaikkapa käytetty aika. Maailmassa/kentässä tulisi hyvä olla myös selvät aloitus- ja lopetuspiste, jotta peli alkaa ja loppuu jostain. Koska kyseessä tulee olemaan kolmiulotteinen maailma, tulee kameran kuvakulman olla sellainen, että kolmiulotteisuus tulee ilmi. Kuvakulma voi olla esimerkiksi takaapäin niin sanotusta kolmannesta persoonasta, jolloin kamera seuraisi ohjattavaa hahmoa.

Liikkeentunnistussensoriin minulla ei ole niin suuria kriteereitä kuin kiihtyvyyssensorille. Tavoitteenani on vain luoda esimerkkikokonaisuus, josta käy ilmi miten gyroskooppia kannattaa Windows -puhelimille ohjelmoitaessa lähestyä. Tämä käy ilmi minusta hyvin, jo yksinkertaisella valmiilla esimerkillä, jossa gyroskoopilta saadut tiedot näkyvät älypuhelimen näytöllä tekstinä. Lähtökohtaisesti gyroskooppia on myös astetta vaikeampi lähestyä, koska se on uudehko sensori, eikä emulaattori tue sitä laisinkaan. Tämän lisäksi Lumia 800:ssa ei sitä ole myöskään, joten en pystynyt sitä itse konkreettisesti testaamaan. Gyroskooppiesimerkki onkin siis vain suuntaa antava kokonaisuus, eikä toimiva demo, niin kuin kiihtyvyyssensorilla. Aloitan isommasta kiih-

tyvyyssensori demosta, jonka jälkeen keskityn gyroskooppiin. Gyroskooppidemo pohjautuu Microsoftin tekemään valmiiseen esimerkkiin.

4.1 Kiihtyvyyssensoridemo

Kaikkein tärkeintä Windows -puhelimille koodattaessa Microsoft Visual Studiota käyttäen, on muistaa ladata sille XNA Game Studio 4.0 ja Windows Phone 7.1 kehitys ympäristö, jotta kaikki seuraavat vaiheet olisivat edes mahdollisia. Ensimmäiseksi kiihtyvyyssensoriin pohjautuvassa pelissä, kun projekti on ensin luotu, on kaikki käytettävissä olevat sensorit tuotava XNA:n käyttöön ulkopuolisesta kirjastosta. Tämä tapahtuu lisäämällä ”using”:n avulla koodiin kuvassa 14 näkyvällä tavalla Microsoftin laitteistopohjainen sensorikirjasto. Tätä ennen kuitenkin pitää muistaa lisätä projektivalikosta ”Solution Explorer” kyseinen kirjasto ”References” kohdasta, jotta sitä voidaan edes käyttää koodin puolella. Sieltä löytyy ”add Reference”:n alta samanniminen kirjasto kuin kuvan 14 yhteydessä näkyvä sensorikirjasto.

```
using Microsoft.Devices.Sensors;

namespace Liikkeentunnistusdemo1
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        Accelerometer kiihtSensori;
        public Vector3 kiihtLukija = new Vector3();
```

KUVA 14. Ensimmäinen vaihe

Tämän jälkeen voidaan kiihtyvyyssensorille luoda oma muuttuja, kuten kuvassa 14 ”kiihtSensori” sekä luoda muuttuja kiihtyvyyssensorin datan lukijalle ”kiihtLukija”. Myös muut kiihtyvyyssensorin käytössä tarpeelliset muuttujat on hyvä luoda tai alustaa tässä vaiheessa, kuvan 15 ”KiihtAktiivi” ja ”nopeuskerroin”. Tämän lisäksi on hyvä luoda muuttuja ”IsScreenSaverEnabled”, jonka käyttötarkoitukseen paneudun seuraavaksi.

```
bool kiihtAktiivi;
const float nopeuskerroin = 24f;


public static bool IsScreenSaverEnabled;

public Game1()
{

    Guide.IsScreenSaverEnabled = false;

    graphics = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";

    graphics.SupportedOrientations = DisplayOrientation.LandscapeLeft;
}
```

KUVA 15. Vinkkejä käyttömukavuuteen

Kuvan 15 ominaisuudet tekevät Windows -puhelimen kiihtyvyyssensoriin pohjautuvista ohjelmista tietyssä tapauksissa mukavampia käyttää. Tämän takia pidän niitä tärkeänä osana demoani, vaikka ne eivät suoranaisesti vaikuta kiihtyvyyssensorin toimintaan. Kuvassa 15 ensimmäisenä määritetään näytönsäästäjä XNA:n ”Guide” -kirjastosta löytyvän ”IsScreenSaverEnabled” -ominaisuuden avulla yksinkertaisesti, joko määrittelemällä se todeksi ”true” tai vääräksi ”false”. Vääräksi määrittelemällä saadaan näytönsäästäjä ohjelmasta kokonaan pois, ja näin ollen yhtäkkiä päälle menevä näytönsäästäjä ei pääse häiritsemään, tässä tapauksessa pelidemon käyttökokemusta.

Seuraavassa kohdassa pakotetaan näytön kuva pysymään tietyssä asennossa, eikä se siis käänny puhelinta kallistellessa. Se saadaan aikaan grafiikkamanageri luokan kautta määrittelemällä näytön asento haluttuun suuntaan, tässä tapauksessa vasempaan eli ”LandscapeLeft”. Tämä on kätevä ominaisuus varsinkin tietyntyyppisissä kiihtyvyyssensoriin pohjautuvissa peleissä, joissa halutaan että käyttäjälle ei aiheudu kääntelystä turhia kuvan kääntymisiä ylösalaisin kesken pelin.

```
base.Initialize();
kiihtSensori = new Accelerometer();

try
{
    kiihtSensori.Start();
    kiihtAktiivi = true;
}
catch (AccelerometerFailedException e)
{
    // Kiihtyvyyssensori ei voinut käynnistyä
    kiihtAktiivi = false;
}
catch (UnauthorizedAccessException e)
{
    // Emulaattori ei tue kiihtyvyyssensoria
    kiihtAktiivi = false;
}
```

KUVA 16. Sensorin käynnistys

Kuvassa 16 käynnistetään kiihtyvyyssensori ”try”:n sisällä, jotta tiedetään onko se käynnistynyt oikein. Tämä tapahtuu yksinkertaisesti vain muodostamalla ”kiihtSensor”:lle uuden ilmentymän ja antamalla ”kiihtSensor”:lle ”Start” -käskyn, kuten kuvasta 16 voidaan huomata. Sensorin käynnistyminen tarkistetaan antamalla ”kiihtAktiivi” -muuttujalle arvo tosi eli ”true”, jos se on käynnistynyt. Jos kiihtyvyyssensori ei käynnisty tai XNA:ssa käytettävä puhelinemulaattori ei tue kiihtyvyyssensoria, annetaan muuttujalle poikkeuksissa arvo väärä eli ”false”. Tämän tiedon avulla voidaan halutessa ilmoittaa käyttäjälle asianmukainen virheilmoitus. Tuen tarkistaminen on kuitenkin periaatteessa nykyään pelkkä muodollisuus, koska jos Windows phone 7.1 OS paketti on asennettu oikein Visual Studioon, sen pitäisi kiihtyvyyssensoria tukea. Erikseen on olemassa vielä tarkistus tukeeko käyttäjän puhelin kiihtyvyyssensoria, koska kuitenkin kaikki uusimmat älypuhelimet tukevat kyseistä sensoria, koin sen turhaksi. Sitä voisi kutsua jo älypuhelinten vakiovarusteeksi.

```
    // Luodaan tapahtuman käsittelijä kiihtyvyyssenosrille.
    kiihtSensori.ReadingChanged +=
        new EventHandler<AccelerometerReadingEventArgs>(SensoridatanMuutos);

}

public void SensoridatanMuutos (object sender, AccelerometerReadingEventArgs e)
{
    kiihtLukija.X = (float)e.X;
    kiihtLukija.Y = (float)e.Y;
    kiihtLukija.Z = (float)e.Z;
}
```

KUVA 17. Tapahtuman käsittelijä

Kuvassa 17 luoadaan kiihtyvyyssensorille tapahtumankäsittelijä sensorista saadun datan muutoksille. Tämän jälkeen tapahtumakäsittelijän avulla saadut kiihtyvyssensorin X, Y ja Z arvot palautetaan luodun ”SensoridataMuutos”metodin sisällä ”kiihtLukija” muuttujalle. Pitää myös muistaa muuttaa tapahtumankäsittelijältä saadut arvot muuttujalle sopiviksi eli ”float” -luvuiksi.

```
protected override void Update(GameTime gameTime)
{
    if (kiihtAktiivi && kiihtSensori!= null )
    {
        //Tänne kiihtyvyyssensoriin perustuva toiminta
        paikkaZ += kiihtLukija.Y * nopeuskerroin;
        //  Tarvittaessa voidaan määrittää liikkeet jokaiselle akselille haluamallaan tavalla
        //paikkaX += -kiihtLukija.Y  * nopeuskerroin;
        //paikkaY += kiihtLukija.X  * nopeuskerroin;
    }
```

KUVA 18. Sensorin toiminnallisuus

Pelin ”Update” -osioon luodaan kaikki sensorien toiminnallisuus. Tässä tapauksessa pallon paikka määritetään siten, että sen nykyiseen arvoon lisätään kiihtyvyyssensorilta saatu luku muuttujan ”kiihtLukija” avulla ja kerrotaan halutun suuruisella muuttujalla ”nopeuskerroin”, kuten kuvasta 18 voidaan havaita. Kerroin on tärkeässä osassa, koska kiihtyvyyssensorilta saadut luvut ovat miinus yhden ja plus yhden välillä eli erittäin pieniä, ja haluttu muutos kappaleen paikassa tämän takia minimaalinen. Lopullisessa demossa toiminnallisuus Toiminta on hyvä laittaa if-lauseen sisälle, jonka avulla toimintaa ei suoriteta, jos kiihtyvyyssensori ei toimi oikein, katso kuvan 18 if-lause. Viimeisessä versiossa demoani en tarvinnut kuin Z-akselin liikettä, siksi vain se määritellään kuvassa 18, mutta minkä tahansa akselin toiminnallisuus voidaan määrittää ”Update” -osiossa.

```
effect.World = this.mallinosat[mesh.ParentBone.Index] *
           Matrix.CreateScale(1f, 1f, 1f) *
           Matrix.CreateTranslation(new Vector3(this.paikkaX, this.paikkaY, this.paikkaZ));
```

KUVA 19. Sensorin toiminnallisuuden liittäminen objektiin

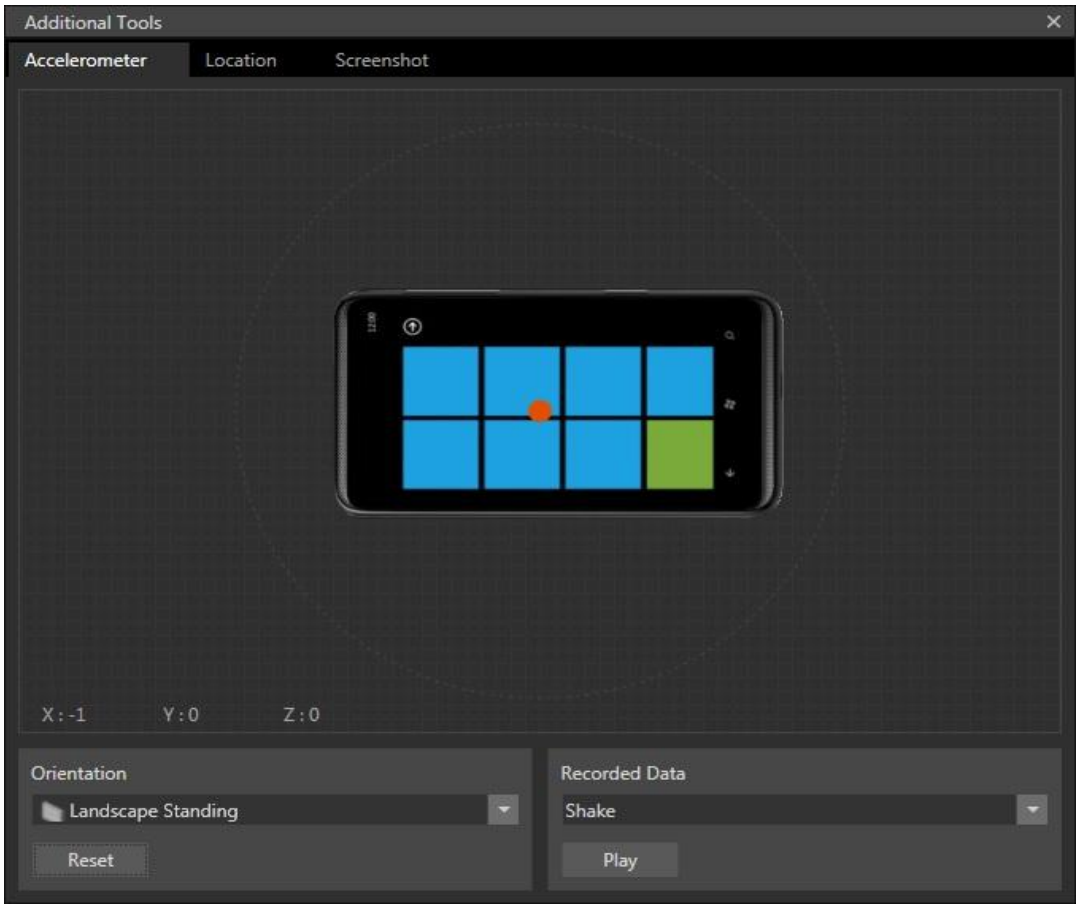
Liikutettava objekti eli tässä tapauksessa pallo, piirretään koodiin halutulla tavalla, kuten kuvassa 19. Jos kiihtyvyys sensorin liikkeet halutaan objektiin, tässä tapauksessa 3D-palloon, täytyy aikasemmin määritellyt muuttujat laittaa 3D-pallon sijainti koordinaateiksi. Tässä tapauksessa ne ovat ”paikkaX”, ”paikkaY” ja ”paikkaZ”, joista kaksi ensimmäistä havanoillistettiin kiihtyvyyssensorin toiminnallisuutta määrittävässä kuvassa 19.

```
    // Poistutaan takaisin nuolella
 if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
    {
        if (kiihtAktiivi)
        {
            try
            {
                kiihtSensori.Stop();
            }
            catch (AccelerometerFailedException e)
            {
                // Kiihtyvyyssensori ei voitu pysäyttää.
            }
        }
        this.Exit();
    } // if
```

KUVA 20. Pelistä poistuminen ja sensorin sammuttaminen

Tärkeä vaihe sensoreita käytettäessä on mielesätäni myös muistaa sammuttaa ne, että ne eivät turhaan kuluta puhelimen akkua ja resurseja esimerkiksi kun peli on pysäytettynä tai valikossa. Kuvassa 20 on kiihtyvyyssensorin pysäytys laitettu jokaisessa Windows -puhelimessa löytyvään kiinteään takaisin-nappiin, ja samasta napista myös poistutaan demosta. Yhtä hyvin kyseessä voisi kuitenkin olla pelin sisäinen nappula näytöllä, jota painamalla pystyttäisiin pysäyttämään peli tilapäisesti ja samalla tavalla sammutettaisiin kiihtyvyyssensori.

Pysäytys hoituu antamalla ”kiihtSentor”:lle ”Stop”-käsky ja laittamalla se try-catch-lauseen kanssa kahden sisäkkäisen if-lauseen sisään. Ylemmässä if-lausessa määrittäen mitä painetaan, ja toisessa ”kiihtAktiivi” muuttujan avulla onko kiihtyvyyssensori toiminnassa. Try -osassa suoritetaan käsky ja catch-osassa voidaan tarkistaa onko kiihtyvyyssensori pysähtynyt ja antaa tarvittaessa käyttäjälle siitä viesti, katso kuva 20.

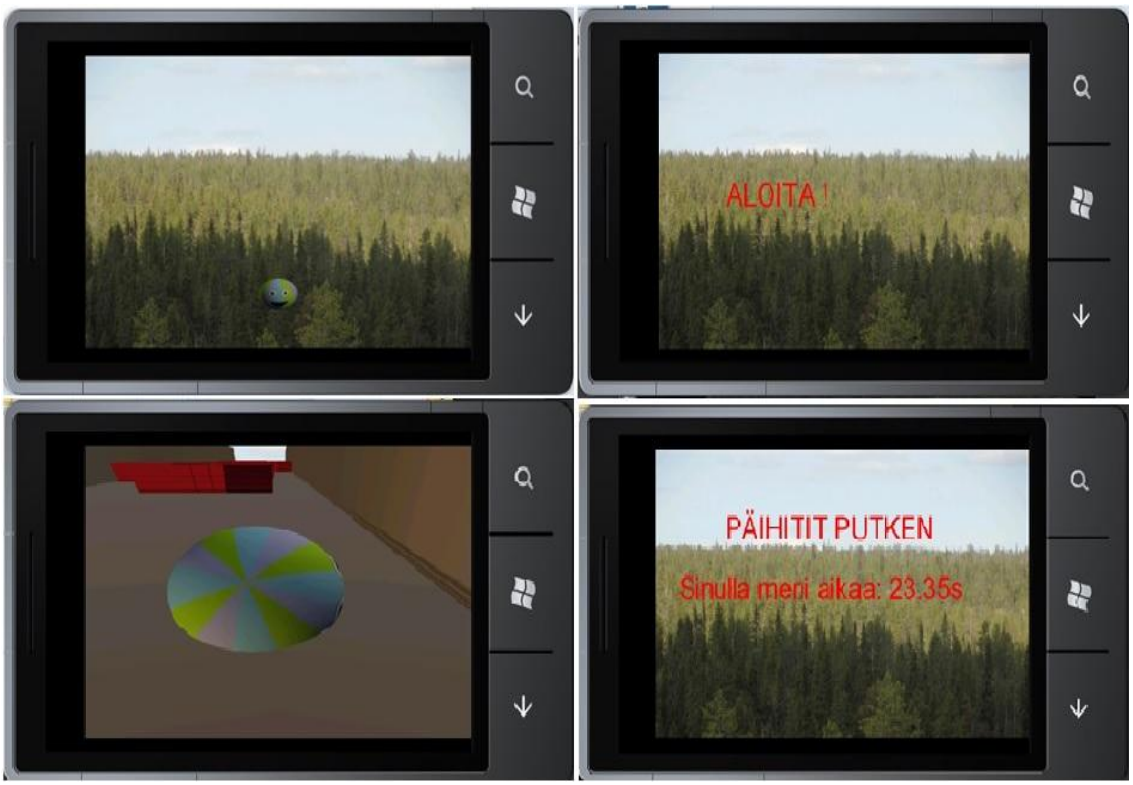


KUVA 21. Kiihtyvyyssensori emulaattori Visual Studiossa

XNA tarjoaa Windows -puhelimen emulaattorin lisäksi mukavan ja helpon tavan testata niin GPS:ää kuin kiihtyvyyssensoria. Kuvassa 21 on näkymä emulaattorin yhteydessä olevasta komponentista, jonka avulla puhelinta pystyy liikuttelemaan virtuaalisesti lähes kuin oikeaa puhelinta eli kiihtyvyyssensoriemulaattori. Valittavana on myös puhelimen täristys toiminto, jonka avulla voidaan testata miten kiihtyvyyssensori reagoi täristykseen.

Tärkeintä on kuitenkin huomioida kuvan 21 alalaidassa näkyvä vetovalikko, josta voidaan valita missä asennossa puhelin virtuaalisesti on, sillä on myös väliä onko puhelin vaaka vai pysty asennossa maahan nähden. Testaaja huomaa tämän heti käytännössä, koska vasemman alalaidan X,Y ja Z arvot eivät pysy vakiona vaan muuttuvat. Tämän huomasin aiheuttavan konkreettisia ongelmia, jos puhelinta kääntelee. Esimerkiksi, jos ohjattavuus oli tehty huomioiden vain vaakatasossa pystyssä olevaa puhelinta, ohjattavuus toimi täysin väärin, jos puhelinta piti normaalissa pystyasennossa. Tämä ongelma tulee huomioida, jos ohjelma vaatii, että liikkeentunnistus toimii samalla tavalla asennosta riippumatta. Demossani tämä ei kuitenkaan ole tarpeellista, sillä optimaalinen peliasento on pelata sitä vaakatasossa pystyssä, kuten emulaattorin kuvassa 21. X,Y ja Z arvojen muuttumisen joutuu myös ottamaan huomioon esimerkiksi kolmiulotteista labyrinttia tehdessä, koska valitut liikkumis-suunnat pysyvät vakiona, vaikka kamerakulma muuttuisi. Tästä seuraa se, että luontevasti esimerkiksi pelitilanteessa eteenpäin mentäessä, puhelinta joutuukin kallistamaan alaspäin, niin kuin takaperin mentäessä. Tämän, kuten edeltävänkin ongelman pystyy selvittämään erilaisilla ehtolauseilla, kuten if.lause.

Kuvasta 22 käy ilmi demoni kehitys emulaattori näkymissä. Vasemmassa kulmassa näkyy pelkkä tausta ja 3D-Pallo, jota pystyi demon kehityksen tässä vaiheessa yksinkertaisesti vain liikuttelemaan edestakaisin puhelimen ruudulla. Tästä kehitin sen jälkeen sitä kuuluisaa lihaa luitten päälle, jonka näkee kuvan 22 muista ruuduista. Oikeassa yläkulmassa on emulaattorin näkymä tekemäni pelidemon alkuruudusta. ”Aloita” tekstiin on laitettu kosketuksen tunnistus tietylle alueelle, jolloin sitä painamalla peli alkaa. Kyseessä voisi olla, mikä tahansa graafinen elementti, joka toimisi täysin samoin.



KUVA 22. Emulaattorin ruutuja pelin eri vaiheista

Vasemmassa alalaidassa on ruutu itse lopullisesta pelidemon ruudusta. Siinä näkyy ohjattava pallo takaapäin, mikä kulkee ja pyörii automaattisesti eteenpäin kameran seuratessa palloa pallon takana. Tarkoituksena on olla osumatta ruskeisiin seiniin ja punaisiin esteisiin, jotka kaikki ovat kolmiulotteisia. Seiniin tai esteisiin osuttaessa, pallo siirtyy takaisin alkuun. Törmääminen rekisteröidään kolmiulotteisilla törmäyslaatikoilla ja – pallolla. Peli loppuu kun pelaaja pääsee kentän loppuun. Tämän jälkeen ruutuun tulee näkyviin aika, jonka pelaaja kentän läpäisyyn käytti, kuvan 22 oikean alakulman havainnollistamalla tavalla. Tulos näkyy hetken ruudussa, jonka jälkeen peli palaa ”Aloita” -ruutuun.

4.2 Liikkeentunnistussensoriesimerkki

Liikkeentunnistussensoriin pohjautuvaa esimerkkiä eli Gyroskooppidemoa kannattaa lähteä rakentamaan samalta pohjalta kuin aikaisempaa kiihtyvyyssensori demoa, koska niitä lähestytään koodillisesti hyvin samalla tavalla. Aluksi siis kannattaa lisätä ulkopuolinen sensorikirjasto, jonka avulla myös gyroskooppiin ja sen toiminnallisuuksiin päästään käsiksi. Tämä kyseinen toimenpide selitetään tarkemmin jo sivulla 18. Jos ei ole tekemässä suoraa XNA-peliä Visual Studiolla, täytyy gyroskooppisovellusta tehdessä muistaa lisätä sensorikirjaston lisäksi ulkopuolinen ”Micro-

sot.Xna.Framework” -kirjasto, katso kuva 23. Jos gyroskoopin halutaan päivittävän dataa käyttöliittymään, niin tämä on mahdollista erillisellä ”timer”:lla, joka vaatii toimiakseen oman ulkopuolisen kirjaston, mutta tämä ei ole välttämätön gyroskoopin toiminnan kannalta.

```
using Microsoft.Devices.Sensors;
using Microsoft.Xna.Framework;
using System.Windows.Threading;
```

KUVA 23. Gyroskoopin oleellisimmat kirjastot (Microsoft 2013)

Tämän jälkeen olisi suotavaa alustaa ja luoda tarvittavat muuttujat, niin kuin kiihtyvyyssensoria tehdessä. Tärkeimpinä on luoda uusi gyroskooppi muuttuja sekä tarvittavat muuttujat, joihin gyroskoopilta saatu data säilötään. Kuvassa 24 muuttujina toimivat ”currentRotationRate” ja ”cumulativeRotation”, vertaa kuvan 14 ”kiihtLukijaan”.

```
public partial class MainPage : PhoneApplicationPage
{
  Gyroscope gyroscope;
  DispatcherTimer timer;

  Vector3 currentRotationRate = Vector3.Zero;
  Vector3 cumulativeRotation = Vector3.Zero;
  DateTimeOffset lastUpdateTime = DateTimeOffset.MinValue;
  bool isDataValid;
```

KUVA 24. Gyroskoopin oleellisimmat kirjastot (Microsoft 2013)

Gyroskooppia ei ole läheskään jokaisessa älypuhelimessa nykypäivänä, koska sitä tukevat vain uusimmat ja kalleimmat älypuhelin mallit. Hyvänä esimerkkinä toimivat juuri omistamani lumia 800, joka on vain reilun vuoden vanha, mutta ei ollut aivan kalliimmasta päästä, ja näin ollen ei tue gyroskooppia laisinkaan. Kiihtyvyyssensorista poiketen onkin hyvä tarkistaa tukeeko puhelin gyroskooppia. Tämä tapahtuu ”isSupported” -ominaisuuden avulla if-lauseessa, kuten esimerkki kuvassa 25. If-lauseessa siksi, että käyttäjää voidaan informoida halutulla tavalla gyroskoopin puuttumisesta, esimerkiksi tekstikentässä kuten kuvassa 25.

```
// Constructor
public MainPage()
{
  InitializeComponent();
  if (!Gyroscope.IsSupported)
  {
    // The device on which the application is running does not support
    // the gyroscope sensor. Alert the user and hide the
    // application bar.
    statusTextBlock.Text = "device does not support gyroscope";
    ApplicationBar.IsVisible = false;
  }
```

KUVA 25. Gyroskoopin tuen tarkistus (Microsoft 2013)

Kun gyroskooppi käynnistetään, tärkeintä on luoda gyroskoopille uusi ilmentymä samalla tavalla kuin kiihtyvyyssensorille eli tässä tapauksessa ” new Gyroscope();”. Ennen kuin gyroskooppi käynnistetään, on kuitenkin muistettava tehdä tarvittavat laskutoimitukset sille, miten usein gyroskooppi hakee tiedot puhelimesta. Tämä tapahtuu ”TimeBetweenUpdate” -ominaisuuden avulla, mikä löytyy gyroskoopista. Saadut arvot voidaan taas esittää esimerkiksi ”TimeSpan”:n avulla, kuten kuvassa 26. Kuten aikaisemmin totesin, tämä ei kuitenkaan ole välttämätöntä gyroskoopin toiminnalle. XNA:lla gyroskooppisovellusta tehdessä taas ei tarvitse tätä laisinkaan, koska siinä on jo valmiina pelisilmukka. Tärkeämpää olisikin tehdä hyvin samanlainen tapahtumankäsittelijä kuin kiihtyvyyssensorille. Vertaa kuvaa 17 kuvan 26 alalaidan toteutukseen.

```
if (gyroscope == null)
{
  // Instantiate the Gyroscope.
  gyroscope = new Gyroscope();

  // Specify the desired time between updates. The sensor accepts
  // intervals in multiples of 20 ms.
  gyroscope.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);

  // The sensor may not support the requested time between updates.
  // The TimeBetweenUpdates property reflects the actual rate.
  timeBetweenUpdatesTextBlock.Text = "time between updates: " +
  gyroscope.TimeBetweenUpdates.TotalMilliseconds + " ms";
  gyroscope.CurrentValueChanged +=
      new EventHandler<SensorReadingEventArgs<GyroscopeReading>>
     (gyroscope_CurrentValueChanged);                                    }
```

KUVA 26. Ilmentymä ja tapahtumankäsittelijä (Microsoft 2013)

Kuten kiihtyvyyssensori, niin myös gyroskooppi olisi suotavaa käynnistää tai sammuttaa, silloin kuin ohjelma niin tarvitsee. Käynnistäminen on hyvä suorittaa jälleen try-

catch-lauseen sisällä, jotta mahdolliset epäonnistumiset käynnistyksessä voidaan havaita, niin kuin kiihtyvyyssensorilla, kuten kuvassa 27. Pysäytys tapahtuu myös samalla tavalla kuin aiemmin eli Stop ”-käskyllä” if-lauseen sisällä, mikä tarkistaa onko gyroskooppi vielä toiminnassa, katso kuva 27.

```
  try
  {
    statusTextBlock.Text = "starting gyroscope.";
    gyroscope.Start();
    timer.Start();
  }
  catch (InvalidOperationException ex)
  {
    statusTextBlock.Text = "unable to start gyroscope.";
  }
}

  if (gyroscope != null && gyroscope.IsDataValid)
  {
    // Stop data acquisition from the gyroscope.
    gyroscope.Stop();
    timer.Stop();
    statusTextBlock.Text = "gyroscope stopped.";
  }
```

KUVA 27. Gyroskoopin ja ajastimen käynnistys ja pysäytys (Microsoft 2013)

Gyroskoopilta saadut tiedot haetaan aiemmin luodun tapahtumakäsittelijän avulla muuttujille kuten esimerkiksi kuvassa 28 ”currentRationrate” -muuttujalle, vertaa kiihtyvyyssensorin vastaavaan kuvaan 17. Tämän jälkeen saatu data syötetään oikeassa muodossa esimerkiksi tekstikenttiin, kuten kuvassa 28 voidaan huomata. Kiihtyvyyssensorin demossa vastaavasti laitoin saadun datan pelin ”Update” -osiossa pallon paikaksi, kuva 18. Tämän jälkeen laitoin datan käyttöön ”draw” -osassa, jossa sijoitin ne 3D-palloon, katso kuva 19.

```
void gyroscope_CurrentValueChanged(object sender, SensorReadingEventArgs
     <GyroscopeReading> e)
  {
    isDataValid = gyroscope.IsDataValid;

    currentRotationRate = e.SensorReading.RotationRate;
  }

  if (isDataValid)
  {
   currentXTextBlock.Text = currentRotationRate.X.ToString("0.000");
   currentYTextBlock.Text = currentRotationRate.Y.ToString("0.000");
   currentZTextBlock.Text = currentRotationRate.Z.ToString("0.000");
  }
```

KUVA 28. Gyroskoopin data käyttöön (Microsoft 2013)

5 PÄÄTÄNTÖ

Älypuhelinten sensorit ovat jo arkipäivää, ja ne ovat älypuhelinten käyttäjille osa arjen askareita jopa huomaamattamme. Puhumalla puhelimeen korvaa vasten laukaisee läheisyyssensorin toimintaan, kääntelemällä selainta ja kuvia haluttuun suuntaan käytämme kiihtyvyyssensoria. Sensorit tuovat älypuhelimille aivan uusia tapoja toimia, ja niistä on muodostunut älypuhelinten tavaramerkki, eikä puhelinta voida enää mieltää älypuhelimeksi, jos siitä puuttuu tärkeimmät sensorit. Pelkkä suoritinteho ja kosketusnäyttö eivät siis mielestäni riitä antamaan puhelimelle älypuhelin titteliä. Tärkeimpiä älypuhelinten tunnusmerkkejä ovat juuri monikosketus ja kiihtyvyyssensorin tuomat ruudun kääntö ominaisuudet.

Omassa kiihtyvyyssensoridemossani onnistuin mielestäni hyvin ja sain lisättyä mukavasti sisältöä kiihtyvyyssensoriin perustuvan pelini ympärille. Alkuvaiheessa asetetut tavoitteet täyttyivät suhteellisen hyvin, mutta visuaalisen ulkoasun lisäksi näen pelissäni paljon kehittämisen varaa. Peliin ensinnäkin olisi hyvä sisällyttää enemmän kenttiä ja, mitkä olisivat monimutkaisempia, kenties jopa labyrinttimaisia. Tällöin joutuisi pureutumaan enemmän ilmenneeseen kamerakulman vaihtumisen aiheuttaneeseen ongelmaan ohjauksessa, jota käsittelin aiemmin. Nykyisille peleille on myös ominaista, että pelissä on jotain kerättävää ja avattavaa. Pelissä voisi esimerkiksi olla jokaisessa kentässä esimerkiksi timantteja, joita keräämällä saisi avattua erilaisia tekstuureita, naamoja tai hattuja pallolle. Tietenkin pelattava hahmokin voisi olla jokin kehittyneempi, esimerkiksi eläin, joka rullalautailisi. Myös äänet ovat tärkeä osa pelikokemusta, joten niiden lisääminen olisi suotavaa. Tämän lisäksi ohjattavuuteen ja sen herkkyyteen voisi tehdä parannuksia ottamalla huomioon eri peliasennot puhelimella. Jos puhelin taas tukisi gyroskooppia, voisi se käyttää sitä hyödyksi kiihtyvyyssensorin kanssa.

Ohjelmapuolella sensorit ovat mahdollistaneet älypuhelinten käyttämisen monipuolisemmin, ja luoneet uusia innovaatiota sekä myös tuoneet uusia ulottuvuuksia vanhoihin. Parhaiten tämä näkyy pelipuolella, jonne on putkahdellut puhelimen kallisteluun perustuvia pelejä eli pelejä, jotka käyttävät kiihtyvyyssensoria ja gyroskooppia. Esimerkiksi ajopeleihin puhelimilla saa aivan erilaisen tuntuman, kun autoa pystyy ohjaamaan melkein kuin ratilla. Tämän lisäksi on paljon sovelluksia ja pelejä, jotka käyttävät hyödyksi kallistelua, GPS-paikannusta, monikosketusta ja kuvasensoria. Uskoi-

sin, että tulevaisuudessa yhä enemmän tulee sovelluksia, jotka yhdistelevät useita eri sensoreita toimivaksi kokonaisuudeksi, ja ottavat hyödyn Internetistä ja sosiaalisuudesta. Millainen olisi esimerkiksi peli, jossa voisit haastaa reaaliajassa kartalla näkyviä ihmisiä lähiympäristöstä ja kisailu käytäisiin jossain kyseisen kaupungin maastossa Google Street Viewin avulla? Kenties jokaisella maalla tai kaupungilla olisi jokin erikois-ominaisuus tai jokaisella mantereella voisi olla myös oma päävihollinen, joten pelin läpäisyyn joutuisi kiertämään maailmaa.

Nykyisellään on ollut paljon kilpaa älypuhelinten kesken suoritintehoissa ja näytön koossa, mutta raja on tulossa jo vastaan ainakin näytön koon suhteen. Tällä hetkellä suurimmat puhelinten näytöt ovat nimittäin 5-6 tuumaan kokoisia, kun taas pienimmät tablettitietokoneet ovat seitsemän tuumaa. Milloin voidaan enää puhua puhelimesta, jos kännyköiden koot vielä kasvavat? Eiväthän ne enää mahdu taskuun, jos samanlainen kehitys jatkuu. Suoritintehoissa taas tuskin tulee rajaa vastaan, mutta minkälaiset tehot ovat järkevät niin että akun kulutus ei olisi kohtuuton? Uusimpien kännyköiden nelinydin prosessoreilla pärjättäisiin varmasti hyvin tulevaisuudessakin, koska se on mielestäni jo nyt liikaa verrattaessa siihen mitä älypuhelimella yleensä tehdään. Eikö olisi järkevämpää kehittää näitä olemassa olevia tehoja, niin että ne olisivat mahdollisimman virtapihejä ja muutenkin joustavampia. Eikä vain lisätä prosessoreiden tehoja ydinten määrää kasvattamalla ja samalla koettaen parantaa akkuja mitä erilaisimmin keinoin.

Mistä älypuhelinten valmistajat voivat siis jatkossa kilpailla? Tietenkin sensoreiden määrällä, ominaisuuksilla ja laadulla. Tämän kehityksen huomaa jo nyt, nimittäin tulevan uuden Samsung Galaxy S4:n myynnin valttikortit perustuvat juuri uusiin tai uudistettuihin sensoreihin. Galaxy S4:n on muun muassa muokattu monikosketuksentunnistussensoria siten, että se tunnistaa kosketuksen myös hanskat kädessä, puhelimen etukameran kuvasensori taas tunnistaa kasvojen ja erityisesti silmien liikkeet. Tämän lisäksi uusina tulokkaina Galaxy S-sarjaan tuodaan lämpötila- ja kosteussensorit, jotka pystyvät mittaamaan nämä puhelimen ympäristöstä eli ulkoilmasta. Kuvasensoreilla kilpaileminen on myös kiihtynyt jatkuvasti ja niissä on nähty valtavia harppauksia viime vuosina, hyvänä esimerkkinä Nokian PureView puhelimessa käytetty tekniikka.

Älypuhelinten sensoreissa on paljon potentiaalia ja näenkin, että mitä enemmän sensoreita niissä on sitä hyödyllisemmiksi ne tulevat myös erilaisissa tutkimustöissä. Miksi vaivautua ostamaan kalliit mittauslaitteet, jos sijoittamalla sen hetkiseen älypuhelimeen saa kokoon kasan erittäinkin herkkiä ja tarkkoja sensoreita. Älypuhelintahan ei sellaisenaan ole pakko käyttää vaan puhelimen voi purkaa ja ottaa sensorit haluamaansa käyttöön, ja niistä voi rakentaa vaikka oman pienen säähavaintokeskuksen pihamaalle. Sensoreiden laatu älypuhelimissa paranee myös vuosien mittaan, joten pienempiin tutkimuksiin ja havaintoihin ne varmasti riittävät tulevaisuudessa erittäin hyvin. Miksei myös suurempiin, jos niitä osataan käyttää oikein.

Älypuhelinten kilpavarustelu sensoreilla on vasta alkamassa ja uskon, että uudet innovaatiot heijastavat myös positiivisesti muualle teknologiaan kuin vain älypuhelinten komponentteihin. Kuten jo teoriaosuudessa näytin on erilaisia sensoreita alettu hyödyntämään, niin tieteen aloilla erilaisissa tutkimuksissa kuin vaikkapa autoissa. Miten olisi, jos kyseisiä sensoreita voitaisiin käyttää hyödyksi myös urheilussa mittalaitteena? Esimerkiksi keihään heitossa sensorin voisi sisällyttää itse suoritusvälineeseen, mikä sitten mittaisi suoraan keihään tai kuulan asennon, nopeuden ja ajan perusteella miten pitkälle se lensi ja lähettäisi tiedon suoraan koneelle ilman erillisiä mittauksia. Isot yritykset kuten Google, näkevät sensoreissa paljon potentiaalia ja he ovatkin luoneet jo esimerkiksi kuskittoman auton, joka perustuu muun muassa erilaisiin sensoreihin. Näenkin, että sensorit pystyvät tekemään asioista hyvin automaattisia ja helpottamaan elämää, teollisuudesta normaalin elämän tilanteisiin kuten esimerkiksi uutisissa ollut älyvessa sen osoittaa. Kuitenkin automaatiolla on rajansa ja jotkut asiat on vain hyvä pitää ihmisten omissa käsissä. Oikeissa käsissä sensoreiden avulla pystytään kuitenkin mielestäni helpottamaan asioita ja varsinkin liikuntarajoitteisten ihmisten arkea erittäin paljon.

Peliala on myös viime vuosina ollut suuresti kiinnostunut ja kunnostautunut erilaisin liikkeentunnistus ohjaimin. Sonyn liikkeentunnistusohjain hyödyntää gyroskooppia, kiihtyvyyssensoria ja kompassia, kun taas Nintendon aikaisemmin julkistettu Wii Remote -ohjain käyttää hyödyksi vain kiihtyvyyssensoria ja tietynlaista kuvasensoria. Nämä erot ja parannukset Sonyn ohjaimen hyväksi tapahtuivat tekniikan kehityksen ansiosta, ja samana vuonna iPhone 4 julkistettiin uuden sensorin eli gyroskooppi mukanaan.

Sonyn ja Nintendon liikkeentunnistusohjaimista poiketen kiinnostavin ja eniten potentiaalia näen kuitenkin Microsoftin Kinectissä, joka perustuu 3D-kuvasensoriin ja mikrofoniin, mitkä pystyvät tunnistamaan ihmisen liikkeitä ja ääntä. Jo nyt Kinectillä on nähty erilaisia innovaatioita esimerkiksi sokean suunnistamisessa auttaminen. Mihin muuhun Kinectiä voitaisiin käyttää tulevaisuudessa? Mielestäni Kinectillä on potentiaalia toimia indie-pelikehityksen apuna mallintaessa aidon tuntuisia 3D-hahmoja sekä kaapatessa hahmoille liikkeet peliin. Näin ei tarvitsisi välttämättä tuhlata aikaa ja rahaa erilliseen liikkeenkaappaus studioon. Tulevaisuus sensoreilla näyttää mielestäni turvatulta ja valoisalta niin älypuhelimissa, kuin nykyään enemmissä määrin muissa laitteissa autoista pelikonsoleihin. Microsoftin vasta julkistettu uusi konsoli Xbox One tukee myös tätä, koska siinä tulee vakiona mukana uudistettu Kinect-laitteisto, joka mahdollistaa entistä tarkemman ihmisten liikkeiden ja äänen tunnistamisen.

LÄHTEET

Ahonen, Tomi 2013. Final Q4 Numbers and Full Year 2012 Stats for Smartphone Market Shares: Top 10 Manufactures, Top 02 Platforms, Top Installed Bases (Revised Corrected). Communities Dominate Brands Blog. WWW-dokumentti. http://communities-dominate.blogs.com/brands/2013/02/final-q4-numbers-and-full-year-2012-stats-for-smartphone-market-shares-top-10-manufacturers-top-os-p.html. Päivitetty 13.2.2013. Luettu 28.4.2013.

Atmel One Touch Sensor 2010. Farnell. WWW-dokumentti. http://www.farnell.com/datasheets/731627.pdf. Päivityksestä ei tietoa. Luettu 24.3.2013.

Burd, Zach 2011. Need for Speed Undercover for Windows Phone 7 and iPhone. Classic game room. WWW-dokumentti. classicgameroom.com/cgrreviews/ 2011/05/02/need-for-speed-undercover-for-windows-phone-7-and-iphone/. Päivitetty. 2.3.2012. Luettu 21.4.2013

ChipWorks 2010. Teardown of the Apple iPhone 4 Smart Phone. WWW-dokumentti. http://www.chipworks.com/blog/recentteardowns/2010/06/23/silicon-teardown-of-the-apple-iphone-4-smart-phone/. Päivitetty 22.6.2010. Luettu 2.4.2013.

Chitkara, Raman. Chin, Robert A 2013. Mobile Technologies Index Image sensor: Steady growth for new capabilities. PwC Technology Institute. WWW-Dokumentti. http://www.pwc.com/gx/en/technology/mobile-innovation/assets/pwc-mobile-technologies-index-image-sensor-steady-growth-for-new-capabilities.pdf . Päivityksestä ei tietoa. Luettu 20.2.2013.

Conway, Joe, Hillegass, Aaron 2010. iPhone Programming The Big Nerd Ranch Guide. Atlanta: Big Nerd Ranch Inc.

Dong, Chelsea 2012. SmartMouse: TurN Your Smart Phone Into A wireless Multi-touch Magic Mouse. TecNode. WWW-dokumentti. http://technode.com/2012/09/17/smartmouse-turn-your-smart-phone-into-a-wireless-multi-touch-magic-mouse/. Päivitetty 17.9.2012. Luettu 12.3.2013.

Farnell. Sensors for Smartphone and Tablet. WWW-dokumentti. http://uk.farnell.com/smartphone-technology-applications. Päivityksestä ei tietoa. Luettu 25.1.2013.

Flip It! Gyro Game 2010. iTunes. WWW-dokumentti. https://itunes.apple.com/us/app/flip-it-gyro-game/id380368539?mt=8. Päivitetty 27.9.2010. Luettu 13.4.2013.

Good News from Finland 2012. Multitouch vamistaa maailman suurimman kosketus-näytön. WWW-dokumentti. http://www.goodnewsfinland.fi/arkisto/uutiset/ multi-touch-valmistaa-maailman-suurimman-kosketusnayton/. Päivitetty 19.9.2012. Luettu 13.3.2013.

Hornshaw, Phil 2010. Fresh iPhone Gmes for Dec. 22: TurboGrafx-16 Gmebox, Pcket God ChuChu Rocket! updates. Appolicious. WWW-dokumentti. http://www.appolicious.com/articles/4471-fresh-iphone-games-for-dec-22-turbografx-16-gamebox-pocket-god-chuchu-rocket-updates. Päivitetty 22.12.2010. Luettu 23.4.2013.

iBeer 2010. iTunes. WWW-dokumentti. https://itunes.apple.com/app/ibeer-5-beers-coffee!-milk/id283914070?mt=8. Päivitetty 4.1.2010. Luettu. 15.3.2013.

iButterfly. iButterfly Hong Kong.WWW-dokumentti. http://ibutterfly.hk/eng/- how-touse.html. Päivityksestä ei tietoa. Luettu 13.4.2013.

Iliff, James 2012. Why immersive virtual reality is the next generation of gaming: part 1. Kurzweil Blog. WWW-dokumentti. http://www.kurzweilai.net/why-immersive-virtual-reality-is-the-next-generation-of-gaming-part-1. Päivitetty 14.7.2012. Luettu 21.2.2013.

Inven Sense 2010. Farnell. WWW-dokumentti. http://www.farnell.com/ da-tasheets/887151.pdf. Päivitetty 20.5.2010. Luettu 27.3.2013.

Inven Sense 2013. Inven Sense Powers Breakthrough Basketball Analysis Tool From InfoMotion Sports Technologies. WWW-dokumentti. http://www.invensense.com/mems/gyro/documents/articles/022713-invensense-powers-breakthrough-basketball-analysis-tool-from-infomotion-sports-technologies.html. Pävitetty 28.2.2013. Luettu 6.4.2013.

Järvinen, Jani 2012. Windows Phone Sovelluskehitys. Jyväskylä: Sanoma Pro Oy.

Larsen, Rasmus 2012. Touch technology in smartphones explained. FlatpanelsHD. WWW-dokumentti. http://www.flatpanelshd.com/focus.php?subaction=showfull-&id=1348049303. Päivitetty 19.9.2012. Luettu 24.3.2013.

Lee, Wei-Meng 2011. Beginning iOS 5 Application Development. Haboken: Wrox.

Light Meter 2013. Google Play. WWW-dokumentti. https://play.google.com/store/apps/details?id=com.bti.lightMeter&feature=related_apps#?t=W251bGwsMSwxLDEwOSwiY29tLmJ0aS5saWdodE1ldGVyIl0. Päivitetty 16.4.2013. Luettu 20.4.2013.

Mechanical Engineering. Light sensor applications.. WWW-dokumentti. http://www.mechanicalengineeringblog.com/tag/light-sensor-applications/. Päivityk-sestä ei tietoa. Luettu 14.4.2013.

Mehta, Daryush D, Zafiartu, Matias, Feng, Shengran W, Cheyne, Hardold A, Hillman, Robert E 2012. Mobile voice health monitoring using a wearable accelerometer sensor and a smartphone platform. WWW-dokumentti. http://people.seas.harvard.edu/~dmehta/docs/MehtaIEEE-TBME2012%20ACCEPTED%20Mobile%20voice%20health%20monitoring%20using%20a%20wearable%20accelerometer%20sensor%20and%20a%20smartphone%20platform.pdf. Päivityksestä ei tietoa. Luettu 19.3.2013.

Microsoft 2013. How to get data from the gyroscope sensor for Windows Phone. Windows Phone Dev Center. WWW-dokumentti. http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202943%28v=vs.105%29.aspx. Päivitetty 3.3.2013. Luettu 14.4.2013.

MultiTouch 2013. WWW-dokumentti. http://www.multitaction.com/about/photos/. Päivityksestä ei tietoa. Luettu 11.4.2013.
N.O.V.A 2011. iTunes. WWW-dokumentti. https://itunes.apple.com/us/app/n-o-v-a-near-orbit-vanguard/id343596730?mt=8. Päivitetty. 7.12.2011. Luettu 15.3.2013.

OKI 2013. OKI Develops Compact Sound Source Separation (SSS) Microphone Module with Amibent Voice Supressor (AVS) Technology. WWW-dokumentti. http://www.oki.com/en/press/2013/01/z12081e.html. Päivitetty 19.1.2013. Luettu. 15.4.2013.

Owano, Nancy 2012- Toshiba smartphone camera sensor has eye on the future. WWW-dokumentti. http://phys.org/news/2012-12-toshiba-smartphone-camera-sensor-eye.html Päivitetty 28.12.2012. Luettu 12.2.2013.

Prigg, Mark. The tiny chip that can track its location WITHOUT satellites. MailOnline. WWW-dokumentti. http://www.dailymail.co.uk/sciencetech/article-2308196/The-tiny-chip-replace-GPS-satellites.html. Päivitetty 12.4.2013. Luettu 20.5.2013.

Proximity Plus Plus Free 2011. Google Play. WWW-dokumentti. https://play.google.com/store/apps/details?id=com.tapan.PreferencesExample&feature=also_installed . Päivitetty 28.11.2011. Luettu 28.4.2013.

Proximity Screen Off Pro 2013. Google Play. WWW-dokumentti https://play.google.com/store/apps/details?id=com.itsme4ucz.screenoffpro&feature=related_apps. Päivitetty 22.4.2013. Luettu 2.5.2013.

Roivas, Panu 2012. Windows 8:n liikkeentunnistus suurennuslasin alla. http://www.hardware.fi/uutiset/artikkeli.cfm/2012/01/27/windows_8_n_liikkeentunnistus_suurennuslasin_alla. Tom's Hardware. WWW-dokumentti. Päivitetty 27.1.2012. Luettu 27.3.2013.

Schirmer, Maximilian, Höpfner, Hagen. Smartphone Hardware Senssors. Mobile media group. WWW-dokumentti. http://www.uni-weimar.de/medien/wiki/images/Zeitmaschinen-smartphonesensors.pdf. Päivityksestä ei tietoa. Luettu 9.3.2013.

Shah, Agam 2013. Inside Samsung Galaxy S4's face- and ey.tracking technology. Networkworld. WWW-dokumentti. http://www.networkworld.com/-news/2013/031513-inside-samsung-galaxy-s-439s-267745.html. Päivitetty 15.3.2013. Luettu 22.3.2013.

Smart Compass 2013. Google Play. WWW-dokumentti. https://play.google.com/store/apps/details?id=kr.sira.compass. Päivitetty 8.3.2013. Luettu 23.3.2013.

Sports Tracker. Introducing Sports Tracker. WWW-dokumentti. http://www.sports-tracker.com/blog/about/. Päivityksestä ei tietoa. Luettu. 3.4.2013,

Steele, James, Nelson, To 2011. The Andoid Developer's CookBook Building Aplicaions with the Android SDK. Boston: Pearson Education Inc.

Sony 2012. WWW-dokumentti. http://www.sony.net/SonyInfo/News/Press/201208-/12-107E/. Päivitetty 20.8.2012. Luettu 12.3.2013.
ST 2009. MEMS digital output motion sensor ultra low-power high percormance 3-axes " nano" accelerometer. WWW-dokumentti. http://www.st.com/web/en/ resource/technical/document/datasheet/CD00213470.pdf. Päivityksestä ei tietoa. Luettu 14.2.2013.

Technomiz 2012. Transform Your Anrdoid Phone Into Microphone For Your PC. WWW-dokumentti. http://technomiz.com/transform-your-andriod-phone-into-microphone/ Päivitetty 21.8.2012. Luettu. 15.4.2013.

Thompson, Chris, White, Jules, Doughberty, Brian, Albright, Adam, Schmidt, Douglas C. Using Smartphones to Detect Car Accidents and Provide Situational Awareness to Emergency Responders. Institute for Software Integrated Systems. WWW-dokumentti. http://www.dre.vanderbilt.edu/~schmidt/PDF/wreckwatch.pdf . Päivityksestä ei tietoa. Luettu 10.4.2013.

Uj Jaman, Asad 2011. Sensors in Smartphones. Mobile Device Insight. WWW-dokumentti. http://mobiledeviceinsight.com/2011/12/sensors-in-smartphones/. Päivitetty 1.12.2011. Luettu 18.1.2013.

Ultimate Mouse. Negusoft. WWW-dokumentti. http://www.negusoft.com/index.php/ultimate-control/78-ultimate-control/83-ultimate-mouse. Päivityksestä ei tietoa. Luettu 20.3.2013.

Yang, Sarah 2012. Floating robots use GBS.enabled smartphones to track water flow. News Center Berkley. WWW-dokumentti. http://newscenter.berkeley.edu/2012/05/09/floating-sensors-track-delta-water-flow/. Päivitetty 9.5.2012. Luettu 1.3.2013.

Zhou, Zhinan, Zhu, Robert, Zheng, Pei Yang, Baijian 2011. Windows Phone 7 Programming for Android and iOS Developers. Haboken: Wrox.