Das Auto

Matti Nelimarkka

22. toukokuuta 2011

Tiivistelmä

Yksinkertainen lapsille ja lasten mielisille toteutettu ohjelma, missä voi leikkiä pikkuautolla. Pikkuauto on myös vaihdettavissa ja arkitehtuuri sallii monimuotoisen ohjelman tekemisen (esimerkiksi eläimet erilaisilla äänillä, ...).

Teknisesti toteutuksessa on hyödynnetty mobiilijärjestelmien erityispiirteitä, samaan aikaan huomioiden käyttökokemus ja pitäen ohjelmaa mahdollisimman yksinkertaisena.

Sisältö

Kuva 1: Kuvankaappaus ohjelmasta

1 Alusta ja ympäristö

Ohjelma on toteutettu Android-käyttöjärjestelmän versiolla 2.2. Tässä versiossa käytössä on ohjelmointirajapinnan versio (*api level*) 8. Toimiakseen ohjelma vaatii laitteelta kiihtyvyyssensorin, kosketusnäytön sekä kameran. Testausalustana on toiminut Samsung Calaxy S-laite. Toimivuutta muilla laitteilla ei ole testattu.

Ohjelman luontiympäristönä on toiminut Helsingin yliopiston tietojenkäsittelytieteen laitosen Android-harjoitustyö -kurssia. Siksi ohjelma esittää ennenkaikkea funktionaalisen prototyypin sovellusideasta, eikä sellaisenaan ole kaupallistettavissa. Ylioistoharjoitustyön luonteen vuoksi työssä on pyritty huomioimaan ohjelmistokehityksellisiä näkökulmia, kuten tietorakenteiden mielekästä valintaa sekä kokonaisarkitehtuuria.

2 Käyttöohje

Ohjelma on tarkoitettu leikkisäksi viihdykkeeksi lapsille. Tämän takia toimintaa on yksinkertaistettu mahdollisimman paljon ja toiminnallisuuksien määrää on rajoitettu. Kuvassa 1 näemme ohjelman päänäkymän. Ohjelman käynnistyessä näytölle ilmestyy kuva autosta. Liikuttamalla laitetta vasemmalle ohjelma toistaa 'Pruuum' -äännettä, liikuttamalla laitetta oikealla ohjelma toistaa 'Pii pii' -äännettä. Näkyvän kuvan voi vaihtaa painamalla näyttöä noin kaksi sekunttia.

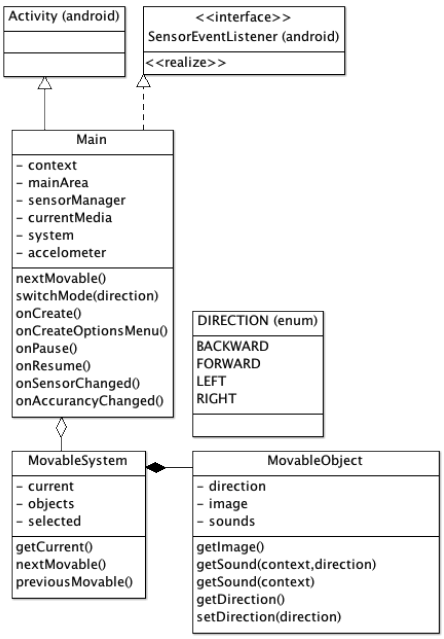
Ohjelmaan on toteutettu myös läpinäkyvä pinta kameran avulla, eli ohjelma kuvaa jatkuvasti taustaansa ja näyttää tämän käyttäjälle. Näin tulisi syntyä mielikuva siitä, että auto liikkuu oikeassa ympäristössä.

3 Ohjelman rakenne

Ohjelman toiminnalllisuudet ovat hyvin yksinkertaiset, minkä johdosta ohjelman rakenne ei myöskään ole poikkeuksellisen suuri. Siinä on pyritty keskittymään mobiilijärjestemän erikoispiirteisiin, eli käytönnössä sensorialustan hyödyntämiseen ja siihen tutustumiseen. Käyttöliittymässä ollaan käytetty yksinkertaisia Android-alustan tarjoamia mahdollisuuksia.

Ohjelma koostuu kolmesta luokasta. Käyttöliittymä on eriytetty `Main`-luokkaan, joka hallitsee myös sensorien tulosten tulkintaa. `MovableSystem` sisältää ohjelman logiikan ja `MovableObject` taas tiedon yksittäisistä liikuteltavista objekteista. Luokkakaavio on kuvassa 2.

`Main`-luokka periytyy Activity-luokasta. Se siis toteuttaa yhden toiminnalllisuuden ohjelmassa. Activitystä periytettyinä metodeina ovat ovat `onCreate(Bundle savedInstanceState)`, `onResume()` sekä



Kuva 2: Ohjelman UML-luokkakaavio

`onPause()`, jotka hallitsevat ohjelman suoritukseen liittyviä vaiheita. Lisäksi `onCreateOptionsMenu(Menu menu)` toteuttaa aukeavan valikon. Liäksi Main-luokka toteuttaa `SensorEventListener`-rajapinnan, joka mahdollistaa sensoreiden käytön.

`MovableSystem`-luokka sisältää tietoa olemassa olevista liikuteltavaa hahmoa. Se tarjoaa käyttöön kolme metodia tähän hallintaa. `nextMovable()` sekä `previousMovable()` muuttavat nyt valittua hahoa ja `getCurrent()` palauttaa nykyisen hahmon.

`MovableObject` sisältää tiedon yksittäisestä liikuteltavasta hahmosta. Hahmolla on suunta, kuva sekä suuntaan liittyvät äänet. in konstruktori `public MovableObject(int image, int forward, int backward, int left, int right)` vaatii parametrinään resurssien tunnisteita (`R`-luokan kautta), joiden perusteella kuva ja äänet määräytyvät. `getImage()`, `getDirection()` sekä `setDirection(DIRECTION direction)` liittyvät tähän. Lisäksi `getSound(Context c)` palauttaa `MediaPlayer`-olion, joka vastaa nykyisen suunnan ääntä.

3.1 Tietorakenteet

Ohjelmassa on käytetty sekä dynaamisia että staattisia tietorakenteita. Jokaisen `MovableObject`in äänet on tallennettu yhteen taulukkoon. Taulukon paikat on määritelty `DIRECTION`-järjestyksen (*enumeration*) kautta, jolloin jatkokehittäjän ei tarvitse näitä tietää.

Vastaavasti kaikki `MovableObject`it on tallennettu `MovableSystem`iin `ArrayList`-rakenteeseen. `ArrayList` on dynaaminen, eli sen koko kasvaa tarvittaessa. Vaihtoehtoisesti tähän tarkoitukseen olisi voinut käyttää kaksisuuntaista linkitettyä listaa, joka olisi tarjonnut paremmat mahdollisuudet saada listan seuraava ja edellinen alkio. Kuitenkin, tässäkin tietorakenteessa olisi ollut haastavaa saavuttaa kierto, joten toteutustavalla ei ole radikaalia eroa.

3.2 Algoritmit ja laskenta

Keskeisin laskennallinen tehtävä ohjelmassa olisi kiihtyvyyden muuttaminen nopeudeksi, jotta voitaisiin tietää, mikä on laitteen suunta. Perusmekaniikkaa on, että $v = \int a(t)\,dt$, eli nopeus on kiihtyvyyden ja ajan integraali. Ohjelmoinnissa sama voidaan toteuttaa yksinkertaistetusti `velocity +=`

`acceleration * deltaT`, missä `deltaT` on aikamittauksien välinen aika. Käytännössä tätä laskentaa ei ole toteutettu, koska jopa normaalissa kiihtyvyyden laskemisessa oli ongelmia.

Keskeinen prosessi liittyy liikkeen tunnistamiseen ja tämän jälkeen tapahtuviin muutoksiin. Askel askeleelta prosessi on seuraava:

1. Havaitse, että kiihtyvyden merkit ovat hetkittäin eri merkkiset.
2. Muuta laitteen suuntaa
3. Keskeytä edellisen musiikkitiedoston soittaminen, jos sellainen on olemassa
4. Käynnistä uuden musiikkitiedoston soittaminen

Android-alustan pitää manuaalisesti hallita tilanteet, jossa ohjelma asetetaan taustalle ja kun ohjelma tuodaan takaisin näkyville. Tähän liittyen algoritmi on seuraava

1. Poista kiihtyvyysanturin tapahtumakuuntelija
2. Keskeytä musiikkitiedoston soittaminen, jos sellainen on olemassa

4 Testaus

4.1 Validointi

Validoinnilla tarkoitetaan, että ohjelmistoratkaisu ratkaisee oikeaa ongelmaa. Tämän projektin tapauksessa ei ole mielekästä pohtia ratkaistavasta ongelmasta, koska kyseisellä ohjelmalla ei ole varsinaisesti reaalimaailman asettamaa tehtävää tai rajoitteita. Sen sijaan on mielekästä arvioida käyttökokemusta (*user experience*), mikä on viihdesovelluksissa merkittävä teema.

Käyttökokemuksella tarkoitetaan nyt lyhyesti käyttäjän, järjestelmän ja ymäristön *context*) muodostamaa kokonaisuutta. Käyttökokemus voi olla sekä hetkittäistä että kumulatiivista, koko järjestelmän käytön aikana syntynyttä kokonaisuutta kuvaava termi. Käyttökokemukselle on useita erilaisia määritelmiä, jotka korostavat eri puolia käyttökokemuksesta[1]. Tässä tapauksessa käyttökokemuksessa helppokäyttöisyys ja leikillisyys (*playful*) olivat keskeisessä asemassa. Käyttökokemuksen mittaamiseen on useita menetelmiä, joista asiantuntija-arviot ja käyttäjätestaukset ovat yleisempiä.

Asiantuntija-arvioiden sijan ohjelmaa testattiin nopeasti yhdellä kaksi ja puolivuotiaalla lapsella. Validoinnin tuloksena oli, että tuttipullo on kiinnostavampi objekti kuin leikkisäksi tarkoitettu tietokonesovellus. Tulosta voi pitää viitteellisenä, koska selvästikään testausajankohta ei ollut sopiva leikkisän kokemuksen testaamiseen.

4.2 Verifointi

Verifioinnilla tarkoitetaan testausta tiettyjen määrättyjen toiminnallisuuksien kautta. Keskeiset toiminnallisuudet liittyvät

[1] Lisää ja tarkemmin, esimerkiksi `http://www.allaboutux.org/`.

Kuvaus tehtävästä	Testauksen tulos
Ohjelman sammuttamisen jälkeen ohjelma on hiljaa	Toimii oletetusti.
Ohjelman on hiljaa kun se on asetettu taustalle	Toimii oletetusti.
Ohjelman taustalta tuomisen jälkeen äänet toimivat jälleen	Toimii oletetusti.
Ohjelman taustalta tuomisen jälkeen kamera toimii jälleen	Toimii oletetusti.

Taulukko 1: Ohjelman käynnistäminen ja sammuttaminen

Android-ohjelman tapauksessa sen toiminnan lisäksi siihen, että ohjelma menee taustalle oikeaoppisesti. Taulukoissa 1 sekä 2 kuvataan testitapaukset sekä testauksen tulokset ohjelman tämän hetkisellä versiolla. Ohjelman toiminnan ainoa ongelma liittyy sensorien tarkkuuteen ja sensoritoteutukseen, mitä tarkastellaan yksityiskohtaisesti kappaleessa 5.1.

5 Arvio toteutuksesta

Ohjelma on toteutettu teknisesti melko hyvin ja mobiilijärjestelmän erityisominaisuuksia: kosketusta, sensoreita ja median toistomahdollisuuksia on hyödynnetty toteutuksessa. Lisäksi Android-järjestelmän toiminnallisuuteen ja rajoitteisiin ollaan tutustuttu.

Kuvaus tehtävästä	Testauksen tulos
Laitteen liikuttaminen vasemmalle aiheuttaa 'Pruum'-äänen	Sensorin tarkkuus ei ole riittävän hyvä huomaamaan selkeästi vasemmalle suuntautuvan liikkeen. Tämän takia laitetta tulee ravistella varsin paljon.
Laitteen liikuttaminen oikealle aiheuttaa 'Pii pii'-äänen	Sensorin tarkkuus ei ole riittävän hyvä huomaamaan selkeästi oikealle suuntautuvan liikkeen. Tämän takia laitetta tulee ravistella varsin paljon.
Näytön painaminen pitkään vaihtaa näytettävän kuvan	Toimii odotetusti.
Kuvat toistuvat kun näyttöä painaa useamman kerran	Toimii odotetusti.
Valikko-näppäimen painaminen avaa valikon	Toimii odotetusti.

Taulukko 2: Ohjelman toiminnallisuudet

5.1 Tunnetut puutteet ja jatkokehitysideat

Ohjelmointirajapinnan versiolla 8 kiihtyvyysanturi on toteutettu siten, että se antaa raakadataa applikaatiokehittäjälle, eli siinä on läsnä maan gravitaation vaikutus. Tästä syystä esimerkiksi pöydällä olevan laitteen kiihtyvyys z-akselin suunnassa on $9.81\frac{m}{s^2}$. Ohjelmointirajapinnan versiolla 9 kiihtyvyysanturista on toteutettu versio, missä annetusta datasta on eliminoitu gravitaation vaikutus.

Yllä kuvattu tilanne tarkoittaa, että kiihtyvyysanturin mittaustulokset ovat virheellisiä tai epätarkkoja. Käytännössä tämä olisi mahdollista korjata ohjelmallisesti, mutta koska seuraavassa Android-järjestelmän versiossa (2.3) on käytössä ohjelmointirajapinnan versio 9, ei ole mielekästä käyttää resursseja tämän ongelman toteuttamiseksi.

Ohjelman leikillisyyttä olisi voinut lisäksi parempaa hyödyntäen kosketusnäytön ominaisuuksia enemmän. Nyt interaktiomallina oleva yksittäinen pitkä painallus olisi voitu korvata vedolla (*swipe*) tai liikesarjalla (*gesture*), mikä olisi voinut lisätä leikillisyyttä.

Tällä hetkellä resurssitiedostot ovat kiinteästi asetettuja, mutta tulevaisuudessa olisi mielekkäämpää tehdä helposti laajennettavissa oleva järjestelmä. Tällöin olisi tarpeen, että yksittäisiä liikuteltavia hahmoja voisi asentaa ja lisätä automaattisesti.