Usporedba načina upravljanja aplikacijom različitim senzorima

Šaravanja, Marijeta

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

USPOREDBA NAČINA UPRAVLJANJA APLIKACIJOM RAZLIČITIM SENZORIMA

Diplomski rad

Marijeta Šaravanja

Osijek, 2016.godina

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 11.07.2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Marijeta Šaravanja
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-615R, 23.10.2013.
Mentor:	Doc.dr.sc. Krešimir Nenadić
Sumentor:	
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Hrvoje Leventić
Naslov diplomskog rada:	Usporedba načina upravljanja aplikacijom različitim senzorima
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Izraditi aplikaciju za Android platformu u kojoj korisnik može upravljati nekim likom na zaslonu. Usporediti način i mogućnosti kontrole lika na zaslonu korištenjem različitih senzora na mobilnom uređaju.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 2 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 2
Datum prijedloga ocjene mentora:	11.07.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

Sveučilište Josipa Jurja Strossmayera u Osijeku
ELEKTROTEHNIČKI FAKULTET OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.07.2016.

Ime i prezime studenta:	Marijeta Šaravanja
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-615R, 23.10.2013.
Ephorus podudaranje [%]:	1%

Ovom izjavom izjavljujem da je rad pod nazivom**: Usporedba načina upravljanja aplikacijom različitim senzorima**

izrađen pod vodstvom mentora Doc.dr.sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj:

1. UVOD

Android je operacijski sustav namijenjen mobilnim uređajima, temeljen na Linux jezgri i razvijen pod okriljem tvrtke Google. Dizajniran prvenstveno za uređaje sa zaslonom osjetljivim na dodir (poput pametnih mobilnih uređaja i tableta) [1], operacijski sustav koristi dodir kao ulaz koji imitira stvarne akcije, poput povlačenja, tapkanja, povećanja i smanjivanja određenog dijela ekrana, te interakcije s objektima na ekranu. Jedan od načina interakcije s objektima je pomoću senzora koje većina Android pokretanih uređaja sadrži, čime se ovaj rad bavi. Senzori pružaju podatke s visokom preciznošću i točnošću, što ima višestruke primjene. Primjerice, igra može pratiti očitanja senzora gravitacije kako bi dovelo do zaključka o korisničkim pokretima, poput nagiba, rotacije ili zamaha.

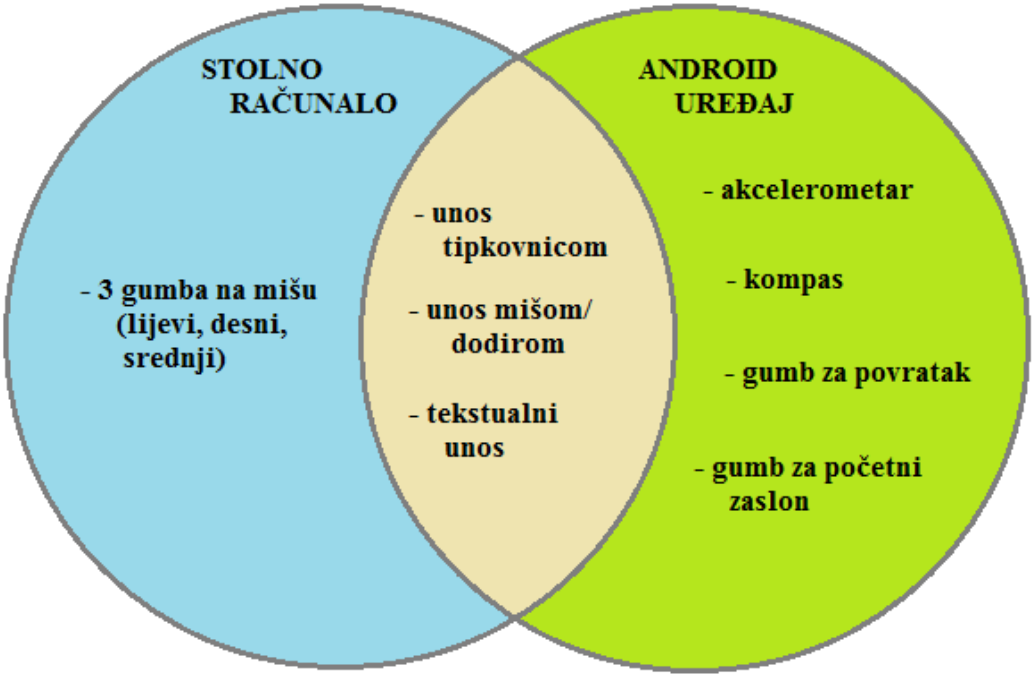
Cilj ovog diplomskog rada opisati je upotrebu različitih senzora za pokretanje određenih objekata na ekranu. U prvom poglavlju je opisana sama upotreba senzora na Android platformi kao i podjela senzora, dok je u drugom poglavlju sadržan detaljniji opis senzora s naglaskom na senzore korištene prilikom same izrade aplikacije. Prikaz implementacije senzora u aplikaciji, kao i slike zaslona unutar same aplikacije su sadržane u zadnjem poglavlju.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je izraditi aplikaciju za Android platformu koja bi sadržavala određenog lika (objekt) na ekranu. Korisnik bi upravljao kretnjom lika na ekranu koristeći različite senzore koje mobilni uređaj sadrži.

2. SENZORI NA ANDROID PLATFORMI

Android uređaji su sposobni samostalno odrediti lokaciju, mjeriti temperaturu, kretanje, otkriti orijentaciju, promjenu u okolini kao i radio signale. Sve to omogućeno je pomoću senzora. Dostupnost senzora čini Android uređaje toliko različitima u odnosu na standardna osobna računala.



Sl. 2.1. *Razlike unosa između osobnog (stolnog) računala i Android uređaja*

Senzori omogućuju, preko aplikacija koje ih koriste, brojne pogodnosti korisniku. Primjerice, spori ručni unosi podataka i manipulacije tih istih podataka su izbjegnuti korištenjem senzora. Jedna od najčešćih upotreba senzora za Android korisnike je upotreba zaslona osjetljivog na dodir, akcelerometra, žiroskopa kao i GPS-a, tj. globalnog pozicijskog sustava (eng. Global Positioning System). Brojne aplikacije koriste akcelerometar kako bi lakše dobile informaciju o tome koristi li korisnik uređaj u vodoravnom ili okomitom položaju. U ovisnosti o informaciji koju senzori proslijede aplikaciji, aplikacija prilagođava prikaz ekrana korisniku (Sl. 2.2.).

Sl. 2.2. *Prikaz prilagodbe aplikacije na mobilnom uređaju načinu korištenja samog uređaja (horizontalni/vertikalni položaj) na temelju očitanja akcelerometra, [2]*

Prema [3], senzor je mogućnost bilježenja mjerenja samog uređaja i njegove vanjske okoline. Ta mogućnost mjerenja ovisi o raspoloživom sklopovlju samog Android uređaja te načinu korištenja. Vrijednosti se mogu direktno prikupljati iz sklopovlja koje mjeri fizičke podatke, poput senzora magnetskog polja. Također je moguće koristiti podatke sklopovlja koje korisnik često koristi, poput kamere i mikrofona. Neovisno o izvoru, dobiveni podaci služe aplikaciji za informacije o stanju uređaja kao i o okolini u kojoj se nalazi. Aplikacije dobivene informacije koriste za razne zadatke, i to u ovisnosti vrste senzora i podataka koji su prikupljeni. Vrste senzora su objašnjene u sljedećem potpoglavlju.

Različite metode unosa koje su omogućene senzorima često pružaju bolji dizajn korisničkog sučelja kao i samu funkcionalnost aplikacija. Senzori su često korišteni od strane programera igara za pametne uređaje. Geste za kontrolu, poput naginjanja pametnog uređaja kao načina unosa, korisne su za korisnike takvih igara jer pružaju izazov i interakciju. Senzori također mogu imati veliku primjenu kod korisnika s fizičkim ograničenjima; primjerice, jednostavnim pokretom bi se omogućilo korištenje različitih aplikacija, unosa teksta i sličnih stvari koje inače nisu moguće korisnicima s određenim ograničenjima.

2.1. Podjela senzora

Senzor je često opisan kao uređaj ili sklopovlje unutar uređaja (pametnog telefona) koje prikuplja podatke iz fizičkog svijeta. To nije u potpunosti točno, budući da postoje senzori zasnovani na sklopovlju (eng. hardware based), ali i programskoj podršci (eng. software based).

2.1.1. Senzori zasnovani na sklopovlju i zasnovani na programskoj podršci

Senzori zasnovani na sklopovlju su fizičke komponente ugrađene unutar mobilnog uređaja ili tableta. Takvim senzorima prikupljeni podaci dolaze od izravnog mjerenja određenih osobina okoline; poput nagiba, magnetskog polja, ubrzanja i slično.

Senzori zasnovani na programskoj podršci oponašaju ponašanje senzora zasnovanih na sklopovlju, iako nisu fizički uređaji, te prikupljaju podatke od istih. Još su nazvani i virtualni ili sintetički senzori. Primjerice, senzorom linearnog ubrzanja je mjereno ubrzanje duž jedne osi. Takav senzor, koji spada u skupinu senzora zasnovanih na programskoj podršci, prikuplja sirove podatke od akcelerometra.

2.1.2. Vrste senzora na Android platformi

Uz tu podjelu senzora, prema [1], Android platforma podržava tri vrste senzora:

- Senzori pokreta

  Senzori pokreta su senzori koji prate kretanje uređaja. Dva takva senzora su zasnovana na sklopovlju - akcelerometar i žiroskop, te tri senzora zasnovana na programskoj podršci – senzor gravitacije, linearnog ubrzanja i vektora rotacije. Senzori zasnovani na programskoj podršci podatke uzimaju od senzora zasnovanih na sklopovlju, ponekad od kombinacije akcelerometra i magnetometra, ponekad samo od žiroskopa. Dostupnost žiroskopa u trenutno postojećim Android uređajima je još uvijek ograničena, dok se akcelerometar nalazi u gotovo svim uređajima.

  Pokreti koje ovi senzori bilježe su primjerice nagib, rotacija ili zamah. Takvi pokreti mogu nastati na dva načina – izravnim pokretom korisnika (npr. zakretanjem uređaja zbog određene aplikacije) ili kao odraz same okoline u kojoj se uređaj nalazi (npr. uređaj unutar automobila u pokretu). Pokreti se bilježe u ovisnosti o točki reference. U prvom slučaju, aplikacija tj. okvir aplikacije predstavlja referencu, dok u drugom slučaju, vanjska okolina predstavlja referencu.

Rezultat mjerenja ovih senzora je izražen kao višedimenzionalno polje. Akcelerometar vraća vrijednosti ubrzanja sile, a žiroskop vrijednosti brzine rotacije za tri koordinatne osi (x, y i z).

- Senzori okoline

  Senzori okoline su senzori koji pružaju informacije o okolini u kojoj se uređaj nalazi. Android uređaji mogu imati četiri takva senzora: senzor temperature zraka, tlaka, vlažnosti i osvjetljenja. Sva četiri senzora su senzori zasnovani na sklopovlju te, osim senzora osvjetljenja koji se nalazi na većini uređaja radi osvjetljenja samog ekrana, njihova prisutnost na Android uređajima ovisi o samim proizvođačima istih. U ovu skupinu senzora se također ubraja i kamera na Android uređaju, budući da prikuplja vizualne informacije o okolini uređaja.

  Uređaji mogu primjerice procijeniti temperaturu u ovisnosti o temperaturi unutarnjih komponenti ili baterije, dok većina uređaja podatke o temperaturi okoline pak uzima od pružatelja usluga prognoze.

  Podaci koje ovi senzori vraćaju kao rezultat ne trebaju nikakva filtriranja ni prilagodbe, stoga se njihov rezultat prikazuje kao pojedinačna vrijednost senzora. Primjerice, senzor temperature vraća vrijednost u Celzijus stupnjevima (°C), a senzor tlaka u hekto Pascalima (hPa).

- Senzori pozicije

  Senzori pozicije su senzori koji određuju poziciju uređaja. Android uređaji imaju dva takva senzora: senzor orijentacije i senzor geomagnetskog polja. Oba senzora su zasnovana na sklopovlju, dok senzor zasnovan na programskoj podršci kojeg se pripisuje ovoj skupini jest senzor blizine. Senzor blizine određuje udaljenost objekta od uređaja, najčešće lica uređaja. Senzor blizine se često uzima u obzir kod programiranja aplikacija, kako bi se odredilo kada je uređaj blizu lica korisnika, primjerice kod dolaznih poziva.

  Rezultat mjerenja senzora orijentacije i magnetskog polja je izražen kao višestruko polje, budući da vraća vrijednosti za sve tri osi koordinatnog sustave (x, y i z), dok senzor blizine kao rezultat vraća pojedinačnu vrijednost.

Potpuni popis dostupnih i podržanih senzora na Android uređajima je prikazan na službenim stranicama posvećenim razvijanju za Android, [1].

2.2. Prikupljanje i upotreba podataka

Kako bi se napravila aplikacija koja koristi senzore, potrebno je identificirati senzore na uređaju kao i njihov kapacitet. Korisno je identificirati sve senzore određene vrste (npr. senzore pokreta) kako bi bio odabran i implementiran onaj koji pruža optimalne rezultate za željenu aplikaciju. Također se preporuča onemogućavanje značajki aplikacije koje bi zahtijevale senzor koji ne postoji na korištenom uređaju. [3]

Nakon što je dostupnost senzora na uređaju utvrđena i određena aplikacija inicijalizirana, dobivaju se očitanja senzora koja je zatim potrebno prikupljati, najčešće koristeći sučelje za programiranje aplikacija (eng. application programming interface, API). Način na koji će ti podaci biti prikupljani ovisi o načinu na koji će ih aplikacija koristiti. Primjerice, aplikacija za lociranje će konstantno prikupljati podatke od senzora, dok će aplikacija za vremensku prognozu svoje podatke prikupljati u određenim vremenskim razmacima.

Očitanja senzora koje je aplikacija prikupila se zatim analiziraju kako bi s njima ostvarili željeni rezultat potreban za daljnju upotrebu. Očitanja određenih senzora se mogu koristiti u njihovom sirovom obliku, dok određeni senzori zahtijevaju algoritme kako bi mogli koristiti njihova očitanja. Primjerice, za senzore pokreta i pozicije često je potrebno koristiti nisko (engl. low-pass) i visoko (eng. high pass) propusne načine filtriranja, kako bi se njihovi podaci mogli koristiti. Ukoliko aplikacija radi sa slikama, najčešće je potrebno smanjiti veličinu slike kako bi se izbjegao negativni utjecaj na memoriju i brzinu obrade. Slika se zatim naknadno obrađuje i izvršavaju se potrebne detekcije, ovisno o aplikaciji.

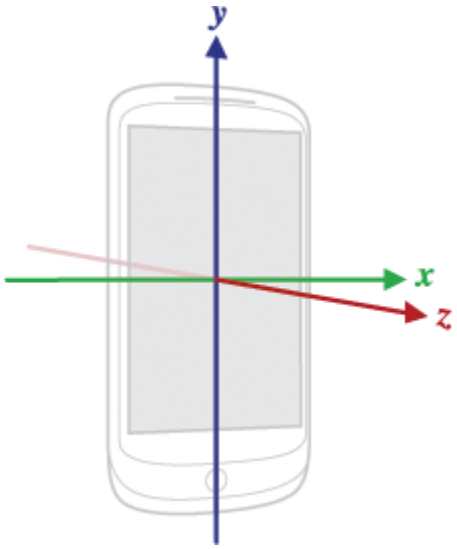
3. SENZORI KORIŠTENI U APLIKACIJI

Dva glavna senzora korištena u aplikaciji su akcelerometar i zaslon osjetljiv na dodir, tj. senzor dodira. Ovo poglavlje će dati detaljniji opis tih senzora.

3.1. Akcelerometar

Akcelerometar mjeri akceleraciju, tj. ubrzanje uređaja duž tri koordinatne osi (x, y i z). To ubrzanje uključuje kako fizičko ubrzanje (promjenu brzine) tako i utjecaj gravitacije.

Generalno, okvir za senzore koristi standardni koordinatni sustav s tri osi za prikaz vrijednosti očitavanja. Za većinu senzora, taj koordinatni sustav je definiran u odnosu na zaslon uređaja kada je uređaj u položaju prikazanom na idućoj slici (Sl.3.1.)



Sl. 3.1. *Koordinatni sustav tri osi (u odnosu na zaslon uređaja) kakav koriste Android uređaji, [1]*

Kada se uređaj nalazi u takvom zadanom položaju, x os je horizontalna i usmjerena je prema desno. Y os je vertikalna os usmjerena prema gore, dok je z os usmjerena prema vanjskoj strani lica uređaja. Takav koordinatni sustav koriste idući senzori [1]:

- Akcelerometar
- Žiroskop
- Senzor gravitacije
- Senzor linearnog ubrzanja

- Senzor geomagnetskog polja

Važno je napomenuti kako je takav položaj uređaja uzet kao zadan položaj. Ukoliko se uređaju promjeni položaj, te uređaj bude postavljen u horizontalan položaj, osi koordinatnog sustava se ne mijenjaju.

Uzevši u obzir da akcelerometar koristi standardni koordinatni sustav, kada uređaj postavimo na ravnu podlogu u zadanom položaju, pomicanje uređaja rezultira na sljedeći način:

- ako je uređaj pomaknut s lijeve strane, tako da se pomiče u desnu stranu, ubrzanje na x os je pozitivno,
- ako je uređaj pomaknut s donje strane, tako da se pomiče prema gore, ubrzanje na y os je pozitivno
- ako uređaj mirno stoji, bez ikakvog micanja, ubrzanje na z os iznosi 9.8 m/s^2

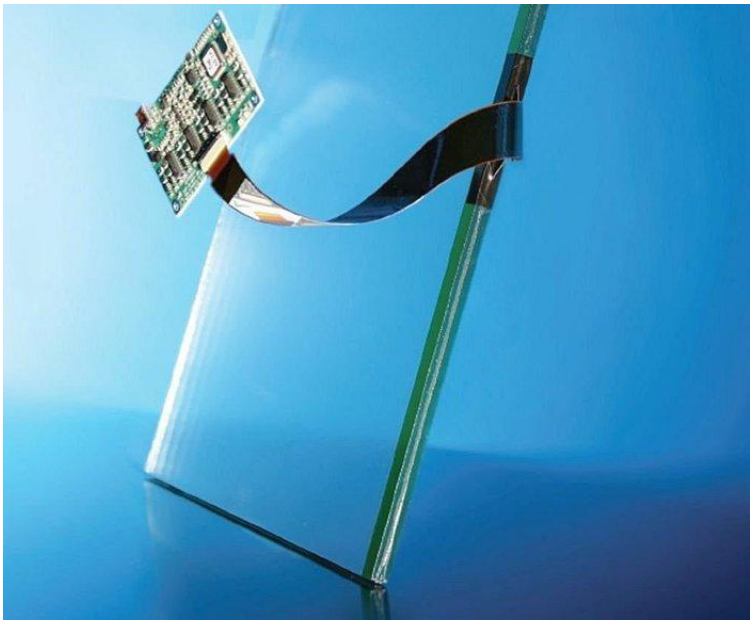
Akcelerometar je često korišten senzor za praćenje kretanja uređaja, te ga imaju gotovo svi Android uređaji. Jednostavan je za korištenje te ima velike primjene u razvoju aplikacija.

3.2. Zaslon osjetljiv na dodir – „senzor dodira“

Većina današnjih pametnih uređaja ima zaslon osjetljiv na dodir. Takav zaslon pruža korisniku dojam izravnog utjecaja na objekte na ekranu. Zaslon, uz samu manipulaciju objekata na ekranu, pruža korisniku interakciju sa samim uređajem. Na takvim zaslonima je također omogućen unos podataka čime su takvi zasloni pristupačniji korisnicima koji nisu pretjerano skloni korištenju standardnih oblika računala, kao što je osobno računalo.

Zaslon osjetljiv na dodir se sastoji od tri osnovne komponente: samog senzora dodira, kontrolera i upravljačkog programa. Senzor zaslona osjetljivog na dodir je u stvarnosti ploča od prozirnog stakla sa površinom osjetljivom na dodir, prikazan na sl.3.2. Ta ploča se stavlja preko zaslona, tako da dio ploče koji reagira na dodir prekriva vidljivi dio zaslona kojeg korisnik koristi. Kontroler povezuje senzor dodira sa računalom, tj. operacijskim sustavom na uređaju. Uzima informacije koje šalje senzor dodira te ih pretvara u informacije koje uređaj može koristiti. Upravljački program govori operacijskom sustavu kako da prevede i obradi informacije koje je kontroler poslao.

Trenutno je na tržištu zastupljeno nekoliko različitih tehnologija senzora na dodir, svaka od njih koristi različite metode za otkrivanje dodira. Tri takve tehnologije, najzastupljenije na mobilnim uređajima, su opisane u nastavku.



Sl. 3.2. *Kapacitivni zaslon osjetljiv na dodir, sa senzorom, [4]*

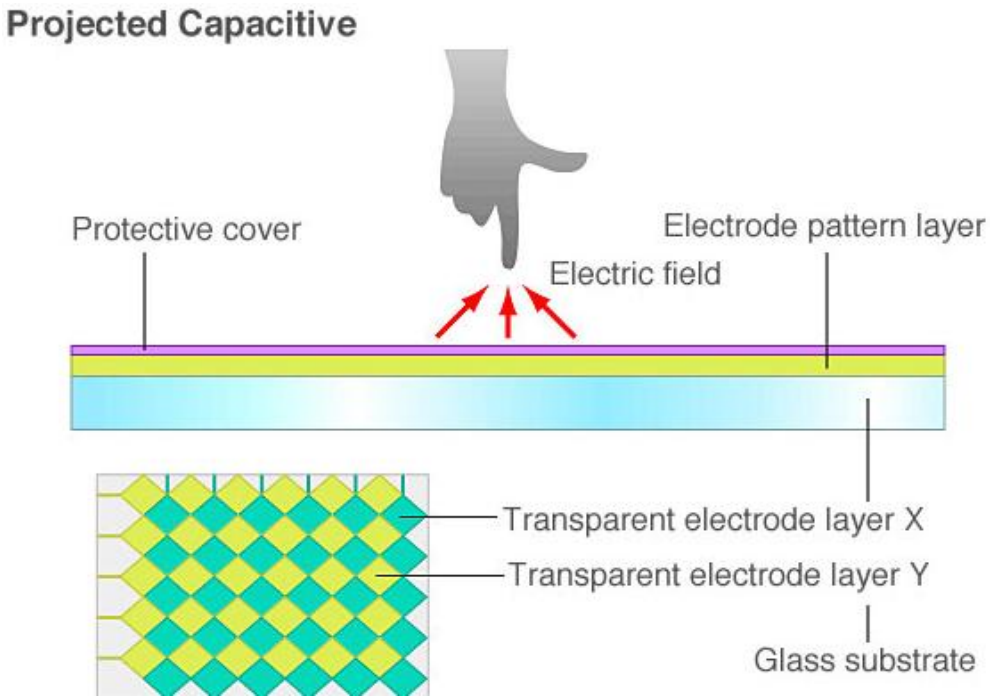
3.2.1. Kapacitivni zasloni osjetljiv na dodir

Kapacitivni zasloni osjetljivi na dodir su, prema [5], druga najčešće korištena metoda za otkrivanje dodira na ekranu. Postoje dvije vrste, objašnjene u nastavku.

- Projicirani kapacitivni zaslon

Projicirani (eng. projected) zasloni su zasloni napravljeni da sadrže jedan otporni sloj koji je pod električnim nabojem. Taj sloj je sastavljen od brojnih tankih žica, koje su isprepletane da tvore rešetku. Na vrhu tog sloja nalazi se uobičajeni sloj načinjen od plastike ili stakla, koji čini zaslon koji korisnik koristi. Kada se korisnik približi prstom zaslonu, elektrostatski kapacitet između više elektroda se mijenja istodobno i ta promjena omogućuje lakše otkrivanje i lokaciju dodira.

Ovakvi zasloni omogućuju otkrivanje dodira nekoliko prstiju (eng. multitouch), te imaju visoku propusnost svjetla, što je prednost kod mobilnih uređaja.

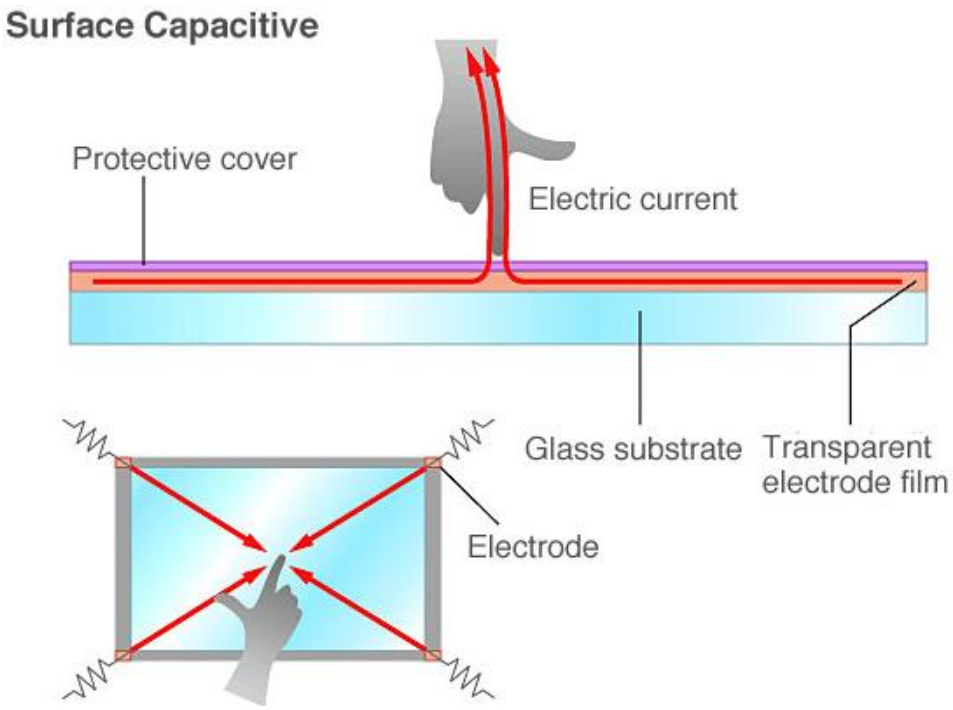


Sl. 3.3. *Prikaz slojeva kod projiciranog kapacitivnog zaslona osjetljivog na dodir, [5]*

- <u>Površinski kapacitivni zaslon</u>

Najčešće se koriste kod vrlo velikih ploča. Prozirni sloj elektroda se nalazi na vrhu staklene podloge, dok se na vrhu sloja elektroda nalazi zaštitni sloj. Sloj elektroda se sastoji od četiri elektrode koje se nalaze u svakom kutu ploče. Elektrode tvore elektrostatsko polje kroz ploču i mjere promjene u polju. Kada korisnik približi prst i dodirne ekran, na mjestu dodira se smanjuje naboj, te dolazi do neravnomjerne izmjene potencijala što elektrode u kutovima bilježe i reagiraju te dolazi do otkrivanja dodira.

Iako ovakva vrsta zaslona ima jednostavniju strukturu od projiciranog kapacitivnog zaslona, te samim time su troškovi manji, ovakav zaslon ne podržava otkrivanje dodira dva ili više prsta.



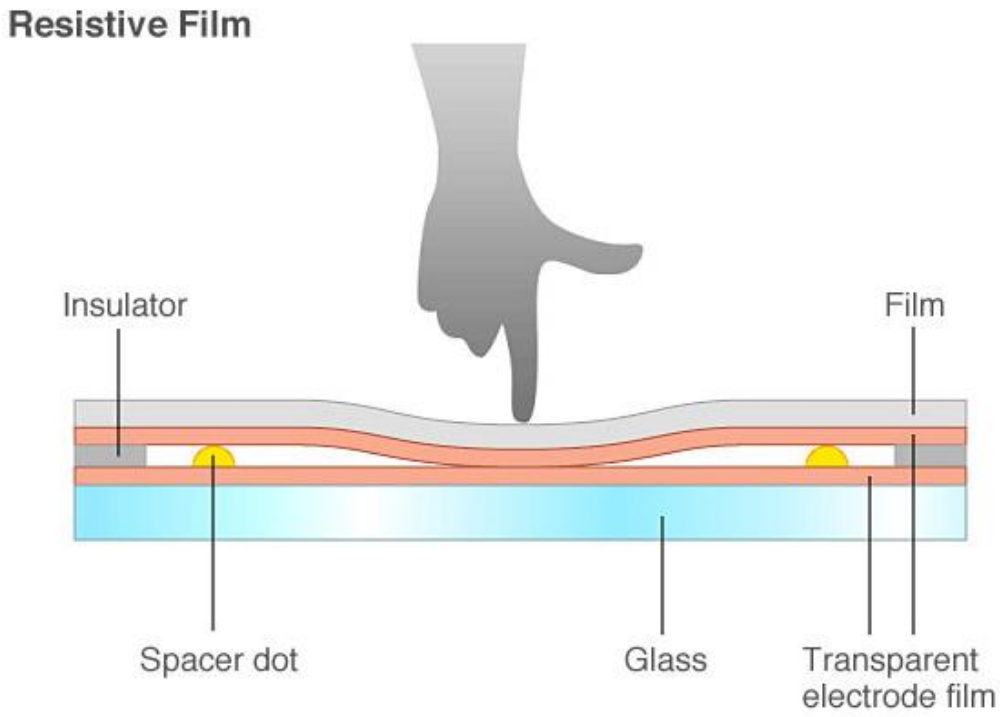
Sl. 3.4. *Prikaz slojeva kod surface kapacitivnog zaslona osjetljivog na dodir, [5]*

3.2.2. Otporni zasloni osjetljivi na dodir

Otporni (eng. resistive) zasloni se nalaze kod brojnih mobitela, GPS uređaja, navigacijskih uređaja u automobila te na igraćoj konzoli Nintendo DS.

Kod otpornog zaslona uski razmak razdvaja dva sloja koja su pod naponom, zbog sloja elektroda koji se nalazi na svakom od njih. Kada korisnik pritisne površinu zaslona, slojevi elektroda se pritisnu i dođu u kontakt, što dovodi do protoka električne struje. To također rezultira promjenom u naponu, što omogućuje kontroleru da izračuna položaj dodira.

Ova tehnologija zaslona osjetljivih na dodir je poprilično jeftina i jednostavna za upotrebu. Budući da je dovoljan samo pritisak na zaslon da bi se otkrio dodir, korisnik može koristiti olovku ili neki drugi predmet kako bi koristio uređaj. Prednost otpornih zaslona nad kapacitivnim je također i preciznost otkrivanja dodira, dok je mana nemogućnost otkrivanja dodira više točaka istovremeno (eng. multitouch).



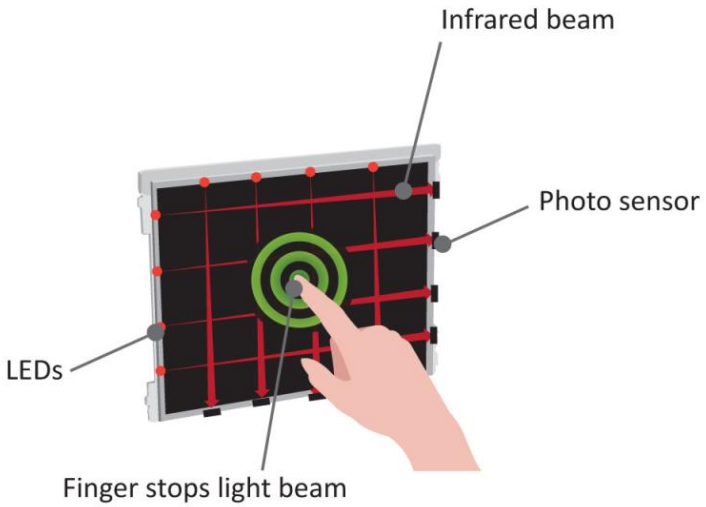
Sl. 3.5. *Prikaz slojeva kod otpornog zaslona osjetljivog na dodir, [5]*

3.2.3. Infracrveni zasloni osjetljivi na dodir

Infracrveni zasloni osjetljivi na dodir funkcioniraju na principu prekida svjetlosne zrake, još nazvano pauza zrake.

Kod ovakvog zaslona, svjetleće diode (LED) su postavljene na jednoj strani okvira zaslona, dok se svjetlosni senzori nalaze na suprotnoj strani, u omjeru jedan na jedan (1:1). Kad korisnik dodirne zaslon, dolazi do prekida svjetlosne zrake, što dovodi do gubitka svjetlosti na senzoru. Budući da su diode i senzori postavljeni u rešetku, svaki dodir razbija dva snopa svjetlosti, što daje x i y koordinate. Te koordinate omogućuju vrlo lagano otkrivanje lokacije dodira.

Zbog načina na koji ova tehnologija radi, korisnik može koristiti infracrveni zaslon dodirima prsta, olovke ili bilo kojih drugih predmeta. Ima veliku propusnost svjetlosti i izdržljiv zaslon.



Sl. 3.6. *Prikaz funkcioniranja infracrvenog zaslona osjetljivog na dodir, [6]*

4. PRAKTIČNI DIO

Praktični dio ovog rada bio je izraditi aplikaciju za Android platformu koja bi sadržavala određenog lika (objekt) na ekranu. Korisnik bi upravljao kretnjom lika na ekranu koristeći različite senzore koje mobilni uređaj sadrži. Aplikacija je izrađena u programskom jeziku Java koristeći aplikacijski okvir libGDX.

4.1. libGDX

LibGDX je besplatni aplikacijski okvir otvorenog izvornog koda namijenjen razvoju igara napisan u programskom jeziku Java uz određene komponente napisane u C i C++ programskom jeziku. Omogućuje razvoj igara za stolna računala i mobilne uređaje koristeći isti kod kao bazu. Aplikacijski je okvir podržan na više operativnih sustava kao što su: Windows, Linux, Mac OS X, Android, iOs i Blackberry.

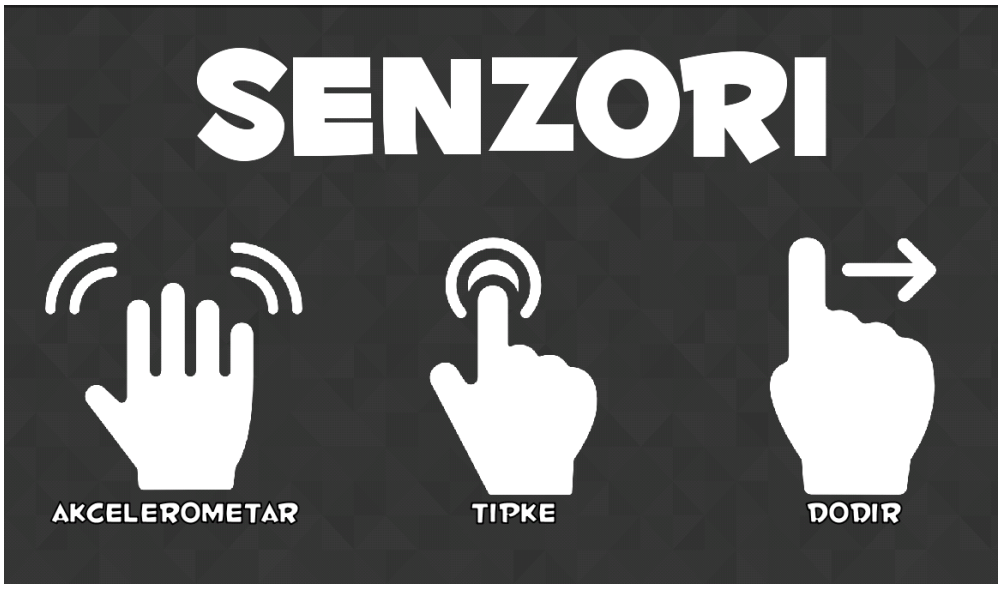
Nastao je sredinom 2009. godine, kao samostalni projekt autora Maria Zechnera, iako pod drugim imenom. LibGDX kao takav je zaživio u travnju 2014. godine.

LibGDX omogućuje programerima pisanje, testiranje i ispravljanje aplikacija na osobnom računalu, a zatim korištenje tog istog koda na Android platformi. Uklanjajući razlike između standardnih Windows/Linux aplikacija i Android aplikacija, libGDX želi osigurati potpunu kompatibilnost između osobnih računala i mobilnih uređaja. [7]

4.2. Razvoj aplikacije

Aplikacija je pisana u Android Studiju, integriranom razvojnom okruženju za razvoj aplikacija za Android platformu. Pisana je u programskom jeziku Java.

Početni zaslon aplikacije prikazan je u na slici 4.1. Tu je sadržana oznaka za naslov, te tri gumba. Pritiskom na prvi gumb, otvara se novi zaslon u kojem je implementiran akcelerometar. Druga dva gumba, i zasloni koji oni otvaraju, koriste zaslon kao senzor. Implementacija tog senzora je izvršena na dva različita načina, što će biti opisano u nastavku.



Sl. 4.1. *Početni zaslon aplikacije „Senzori“*

4.2.1. Implementacija akcelerometra

Implementacija akcelerometra je izvršena kroz primjer nogometne lopte na terenu. Nogometna lopta je željeni lik, a senzor korišten za pomicanje iste je akcelerometar.

Radi lakšeg korištenja i manipuliranja u daljnjem kodu, lopta je definirana kao vlastita klasa. Određena joj je veličina, kao i početna pozicija, te je smještena na sredinu ekrana. (Sl. 4.2. i Sl.4.3.)

```
ball = new Ball();
ballImg = new Texture(Gdx.files.internal("img/fball.png"));
ball.ballSprite = new Sprite(ballImg);
ball.setSize(150,150);
ball.setPosition((Gdx.graphics.getWidth() / 2 -
ball.ballSprite.getWidth() / 2), (Gdx.graphics.getHeight() /
2 - ball.ballSprite.getHeight() / 2));
```

Sl. 4.2. *Kod za deklaraciju lika (nogometne lopte) te postavljanje početne pozicije*

***Sl. 4.3**. Izgled aplikacije sa početno postavljenom pozicijom lika, nogometne lopte*

Aplikacija je postavljena u horizontalni položaj, a dosad je utvrđeno kako se osi ne mijenjaju prilikom promjene položaja uređaja iz okomiti u horizontalni položaj. Radi lakšeg rukovanja podacima i daljnjeg programiranja, varijable za dohvaćanje očitanja akcelerometra na x i y os su zamijenjene, što je vidljivo na slici 4.4.

```
float accelY = Gdx.input.getAccelerometerX();
float accelX = Gdx.input.getAccelerometerY();
float accelZ = Gdx.input.getAccelerometerZ();
```

***Sl. 4.4.** Kod za dohvaćanje očitanja senzora na tri koordinatne osi*

Prilikom stvaranja klase za lik nogometnu loptu, napravljena je metoda za postavljanje pozicije lika, *setPosition().* Ta metoda je potrebna kako bi lopta bila postavljena na novu poziciju ovisno o očitanjima akcelerometra i ovisno o pokretima uređaja. Inicijalno je izvršeno očitanje akcelerometra s uređajem u zadanom položaju, kako bi se stvorila referenca za daljnje programiranje. Nekoliko uzastopnih inicijalnih očitanja za dvije relevantne osi (x i y) su prikazane na slijedećoj slici, sl.4.5.

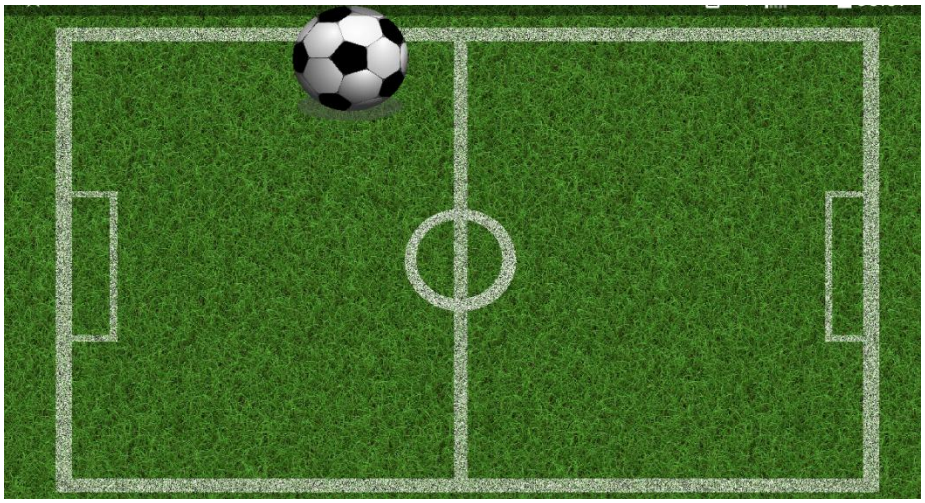
```
y iznosi: 0.038307227
x iznosi: -0.019153614
y iznosi: -0.019153614
x iznosi: 0.0
y iznosi: -0.038307227
x iznosi: 0.0
```

***Sl. 4.5.** Očitanja akcelerometra koja služe kao referentne točke*

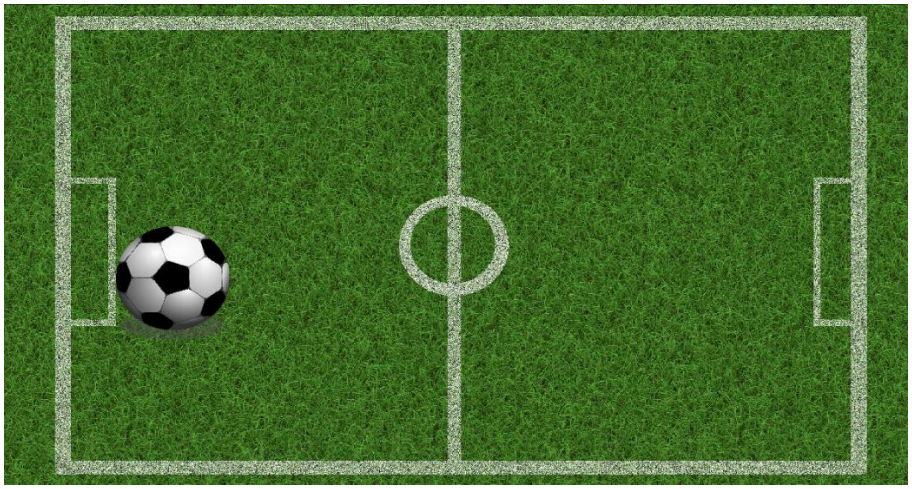
Idući korak je bio napraviti uvjete ovisno o tome u kojem smjeru se odvija kretanje objekta. To je ostvareno koristeći if/else if petlje. Kako bi kretanje bilo linearno s nagibom uređaja, napravljena su tri uvjeta. (Sl.4.6.) Varijable oldX i oldY dohvaćaju stare koordinate objekta, dok varijable newX i newY postavljaju nove koordinate objekta, u ovisnosti o kretanju. Metodom *setPosition()* se spremaju nove koordinate, i postavlja nove pozicija lika. Prikaz kretanja lika je vidljiv na slici 4.7 i slici 4.8.

```
      // kretanje lijevo
        if (accelX < -1.5f && accelX > -2f) {
            newX = oldX - 3;
        }
        else if (accelX < -2f && accelX > -2.5f){
            newX = oldX -4;
        }
        else if (accelX < -2.5f ){
            newX = oldX -5;
        }
       // kretanje desno
        else if (accelX > 2.5f && accelX < 3f) {
            newX = oldX + 3;
        }
        else if (accelX > 3f && accelX < 3.5f) {
            newX = oldX + 4;
        }
        else if (accelX > 3.5f) {
            newX = oldX + 5;
        }
      // kretanje gore
        if (accelY < -1.5f && accelY > -2f) {
            newY = oldY + 3;
        }
        else if (accelY < -2f && accelY > -2.5f) {
            newY = oldY + 4;
        }
        else if (accelY < -2.5) {
            newY = oldY + 5;
        }
       // kretanje dolje
        else if (accelY > 1f && accelY < 1.5f) {
            newY = oldY - 3;
        }
        else if (accelY > 1.5f && accelY < 2f) {
            newY = oldY - 4;
        }
        else if (accelY > 2f) {-
            newY = oldY - 5;
        }

       //nova pozicija
        if (newX > 0 && newX < screenW && newY > 0 && newY <
screenH) {
            ballSprite.setPosition(newX, newY);
        }
```

Sl. 4*Kod postavljanja nove pozicije lika, u ovisnosti o očitanju akcelerometra*



Sl. 4.7. *Prikaz nove pozicije lika kada je akcelerometrom uređaj zakrenut prema gore*



Sl. 4.8. *Prikaz nove pozicije lika kada je akcelerometrom uređaj zakrenut prema lijevo*

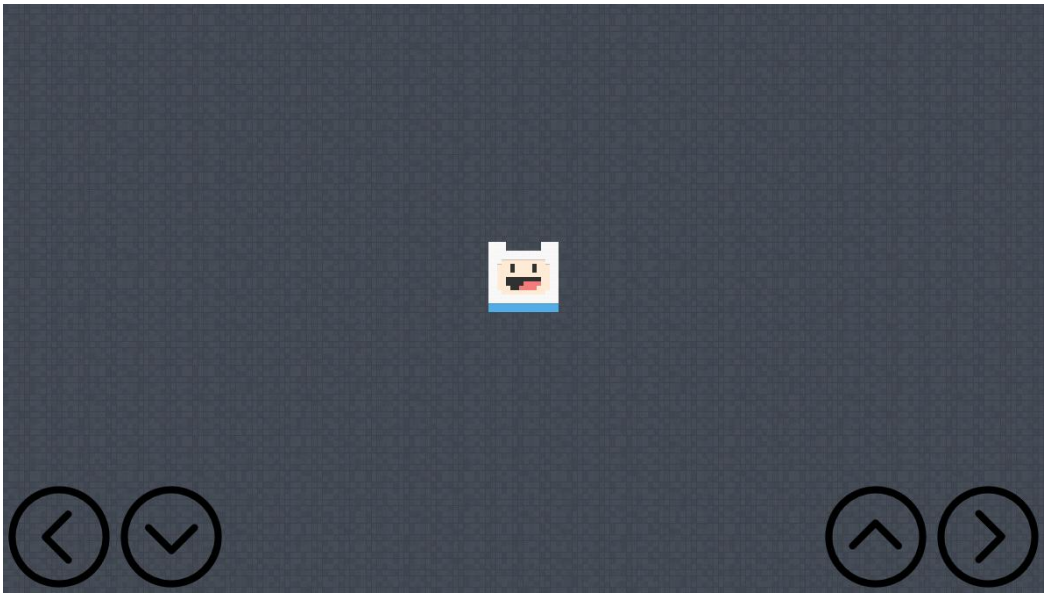
4.2.2. Implementacija zaslona osjetljivog na dodir koristeći gumbe

Iduća implementacija senzora ostvarena je pomoću gumba, koristeći zaslon kao senzor. Primjer ove implementacije je određeni lik koji se kreće kroz mapu.

Stvoren je novi lik kojeg želimo kontrolirati i pomicati po ekranu (Sl.4.9. i Sl.4.10.). Zatim su stvoreni gumbi kako bi mogli na njih, u nastavku programiranja, biti postavljeni slušači (eng. listener).

```
walker = new Walker();
walkerImg = new Texture(Gdx.files.internal("img/finn.png"));
walker.walkerSprite = new Sprite(walkerImg);
walker.setPosition((Gdx.graphics.getWidth() / 2 -
walker.walkerSprite.getWidth() / 2),
(Gdx.graphics.getHeight() / 2 -
walker.walkerSprite.getHeight() / 2));
```

Sl. 4.9. *Kod za deklaraciju lika i postavljanje lika na sredinu zaslona*



Sl. 4.10. *Izgled aplikacije sa početno postavljenom pozicijom lika*

U libGDX-u je za crtanje najčešće korištena klasa glumac (eng. actor), budući da ta klasa sadrži veličinu, boju, poziciju, rotaciju i slično. Za crtanje više glumaca potrebna je pozornica (eng. stage). Pozornica je klasa koja se brine o ulaznim događajima te raspodijeli iste glumcima. Kako bi to bilo moguće, pozornica se mora proslijediti metodi *setInputProcessor()*, što je vidljivo na slici 4.11.

```
Gdx.input.setInputProcessor(stage);
```

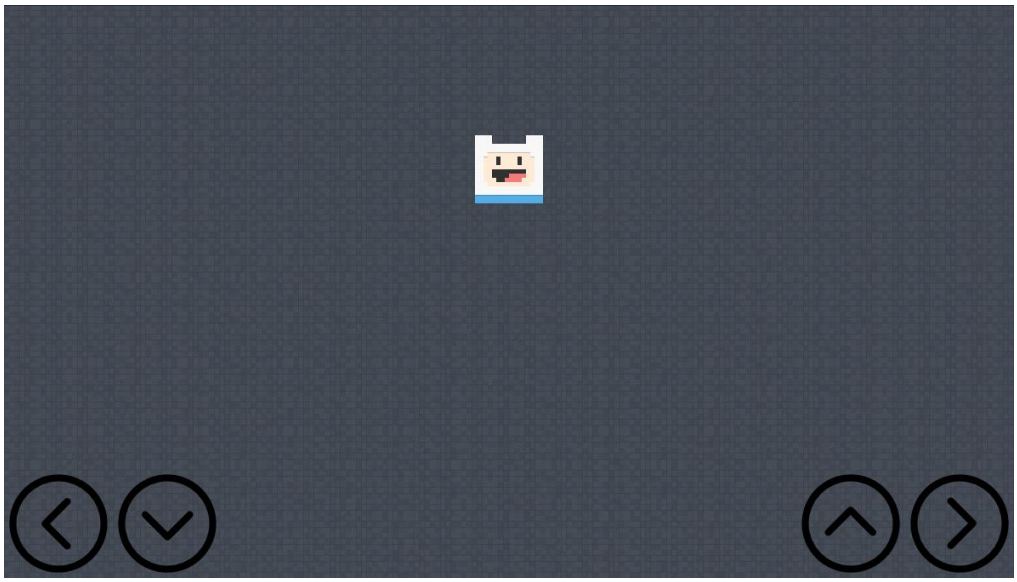
Sl. 4.11. *Kod kojim se pozornica prosljeđuje kako bi se mogla brinuti o ulaznim događajima*

Glumci na pozornici su: glavni lik kojega je potrebno pomicati i gumbi kojima će se upravljati to pomicanje. Na gumbe su stavljeni slušači koji cijelo vrijeme osluškuju za moguće događaje.

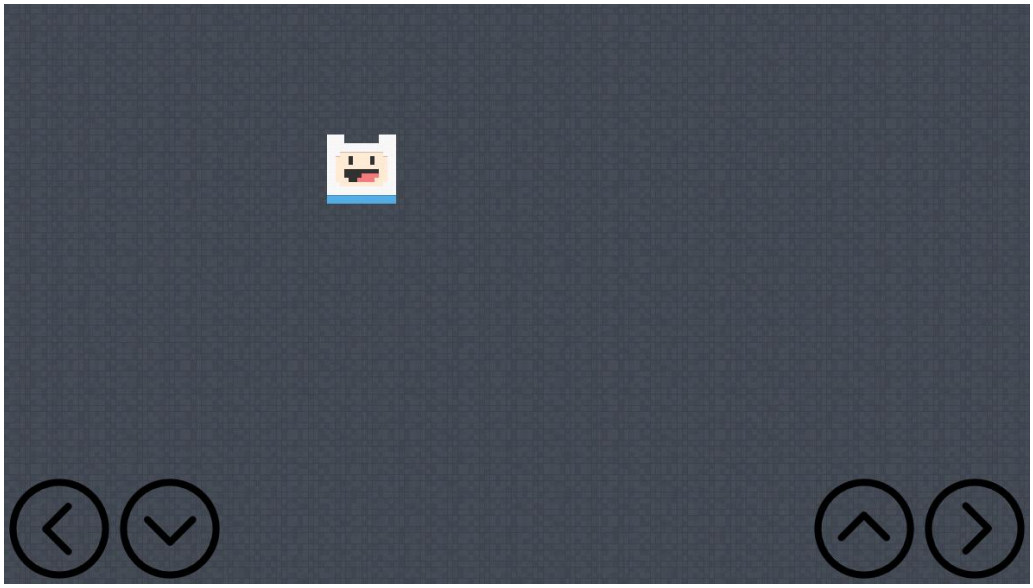
Događaj koji je tu implementiran je dodir prsta, tj. *onClick()*. Kod pojave događaja, u ovisnosti o kojem se gumbu radi, pozivana je sukladna metoda. Slikom 4.12. prikazana je implementacija slušača na dva gumba, kao i njihove odgovarajuće metode, dok je slikama 4.13. i 4.14. prikazana realizacija tog koda.

```
final boolean a = upBtn.addListener(new ClickListener() {
            @Override
   public void clicked(InputEvent event, float x, float y) {
                super.clicked(event, x, y);
                WalkerAssets.moveWalkerUp(x, y);
            }
        });

     public static void moveWalkerUp(float x, float y) {

walker.walkerSprite.setPosition(walker.walkerSprite.getX(),
walker.walkerSprite.getY() + 20);
    }

 final boolean c = leftBtn.addListener(new ClickListener() {
            @Override
   public void clicked(InputEvent event, float x, float y) {
                super.clicked(event, x, y);
                WalkerAssets.moveWalkerLeft(x, y);
            }
        });

   public static void moveWalkerLeft(float x, float y)

walker.walkerSprite.setPosition(walker.walkerSprite.getX() -
20, walker.walkerSprite.getY());
    }
```

Sl. 4.12. *Kod implementacije slušača na 2 različita gumba i metode koje se pozivaju na pojavu događaja*



***Sl. 4.13**. Prikaz nove pozicije lika nakon što je pritisnut gumb za gore*



***Sl. 4.14.** Prikaz nove pozicije lika nakon što je pritisnut gumb za lijevo*

4.2.3. Implementacija zaslona osjetljivog na dodir koristeći povlačenje

Implementacija senzora ostvarena je ponovno koristeći zaslon kao senzor. Primjer ove implementacije je lik psa koji lovi hranu po ekranu. Korisnik ima mogućnost ili dodirom prsta na ekran pomaknuti lik psa što je gotovo istovremena akcija, ili povlačeći prst po ekranu, pomicati lik psa do hrane.

Prvi korak je stvoriti lik psa, napravljen je na način kao do sada. Uz samo stvaranje klase za psa, definirana mu je početna pozicija, koja je na sredini zaslona. (Sl.4.15.)

```
eater = new Eater();
eaterImg = new Texture(Gdx.files.internal("img/dog.png"));
eater.eaterSprite = new Sprite(eaterImg);
eater.setPosition(Gdx.graphics.getWidth() / 2 -
eater.eaterSprite.getWidth()/2, Gdx.graphics.getHeight() / 2
- eater.eaterSprite.getHeight()/2);
```

Sl. 4.15. *Kod za deklaraciju lika i postavljanje lika na sredinu zaslona*

Prilikom ove implementacije senzora, potrebno je omogućiti prihvaćanje ulaznih događaja na čitav zaslon, budući da želimo omogućiti korisniku mogućnost upravljanja likom pritiskom prsta ili povlačenjem prsta po cijelom zaslonu. Radi toga, potrebno je postaviti metodu *setInputProcessor()* na čitav zaslon, što je vidljivo na slici 4.16. Tom linijom koda omogućavamo osluškivanje za sve buduće događaje na cijelom zaslonu.

```
Gdx.input.setInputProcessor(this);
```

Sl. 4.16. *Kod kojim je postavljen InputProcessor na čitav zaslon aplikacije*

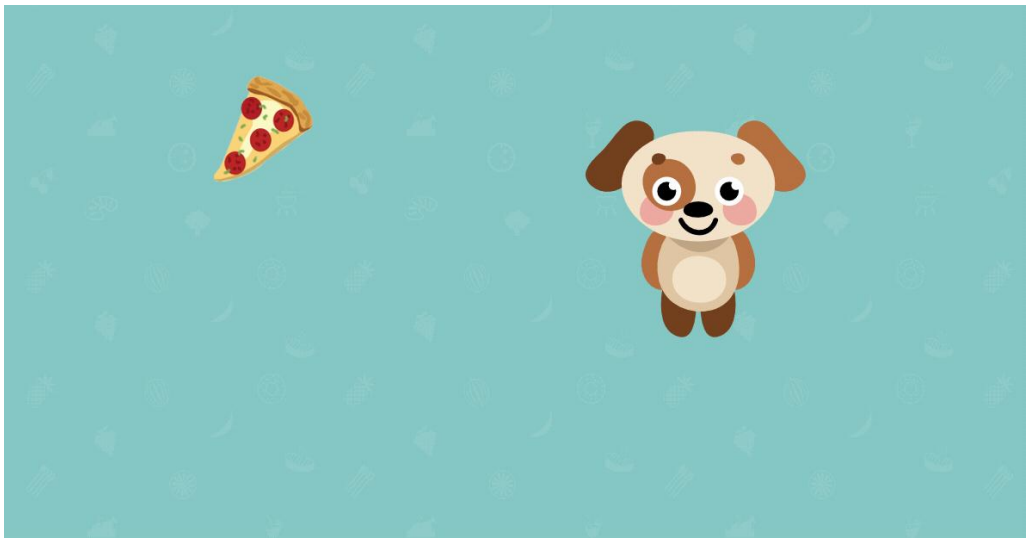
U implementaciji senzora na ovaj način su korištene metode koje dolaze u sklopu sučelja InputProcessor, a to su [8]:

- *touchDown()* - metoda biva pozvana kada korisnik spusti prst na zaslon uređaja, javlja koordinate kao i indeks pokazivača
- *touchUp()* - metoda biva pozvana nakon što je prst podignut sa zaslona uređaja, javlja posljednje poznate koordinate kao i indeks pokazivača
- *touchDragged()* – metoda biva pozvana kad korisnik povlači prst po zaslonu uređaja, javlja koordinate i indeks pokazivača

Implementacija senzora je ostvarena korištenjem metode *touchDragged()*, u kojoj su uzete koordinate senzora kao početne, tj. stare koordinate lika, te su konstantno zapisivane u nove dok god senzor vrši očitanje, odnosno dok god korisnik povlači prst po ekranu. *If()* petlja je korištena kako bi lik ostao u okvirima zaslona, tj. kako nove koordinate ne bi izašle iz koordinata veličine zaslona. Ovaj dio koda je prikazan na slici 4.17.

```
public boolean touchDragged(int screenX, int screenY, int
pointer) {

        float draggedX = screenX;
        float draggedY = screenY;

        newEaterX = draggedX -
EaterAssets.eater.eaterSprite.getWidth() / 2;
        newEaterY = -draggedY +
EaterAssets.eater.eaterSprite.getHeight() +
EaterAssets.eater.eaterSprite.getHeight();

    if (newEaterX > 0 && newEaterX <
(Gdx.graphics.getWidth() -
EaterAssets.eater.eaterSprite.getWidth())
            && newEaterY > 0 && newEaterY <
(Gdx.graphics.getHeight()-
EaterAssets.eater.eaterSprite.getHeight())) {

        EaterAssets.eater.newPosition(newEaterX, newEaterY);
    }
```

Sl. 4.17. *Kod metode touchDragged() za pomicanje lika, dok god postoji pokret povlačenja*



Sl. 4.18. *Vizualni prikaz implementacije zaslona osjetljivog na dodir koristeći povlačenje*

5. ZAKLJUČAK

Cilj ovog rada bio je napraviti Android aplikaciju i koristeći različite senzore omogućiti upravljanje likom na ekranu. Kako bi ostvarenje toga bilo što bolje, potrebno je vidjeti koji su sve senzori podržani na Android platformi, te na koji ih način implementirati.

Android platforma podržava velik broj senzora različitih kapaciteta i različitih namjena. Korisno bi bilo identificirati sve moguće senzore određene vrste (u ovisnosti o potrebi) kako bi odabrali i implementirali upravo onaj koji će pružiti najoptimalnije rezultate za željenu aplikaciju. U radu su implementirana tri senzora, odnosno dva različita senzora, ali tri načina korištenja. Sve implementacije su pružile očekivani rezultat, potpunu kontrolu nad upravljanjem likom. „Tko su budući korisnici aplikacije?“, „Koliko će biti zahtjevna aplikacija?“, neka su od pitanja koja si programeri trebaju postaviti prilikom donošenja odluke o senzorima koje žele implementirati u svoju aplikaciju.

Kao što je rečeno u radu, prvenstveno senzori su ono što razlikuje gotovo sve pametne, a ne samo Android uređaje od standardnih računala. Osim kontrole kretanja likova na ekranu, senzori pružaju pregršt mogućnosti, a te mogućnosti će se povećati još više sa većom rasprostranjenošću senzora koje mogu omogućiti proizvođači pametnih uređaja. Senzori također veliku ulogu igraju kod osoba sa raznim fizičkim ograničenjima, stoga je primjena senzora u svakodnevnom životu pozitivna i vrlo bitna.

LITERATURA

[1] Google, [Mrežno]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html. [Pokušaj pristupa 12. 3. 2016.].

[2] Google, [Mrežno]. Available: https://developer.chrome.com/multidevice/webview/pixelperfect. [Pokušaj pristupa 6. 6. 2016.].

[3] G. Milette i A. Stroud, Professional Android Sensor Programming, Indianapolis, Indiana: John Wiley & Sons, Inc., 2012.

[4] Crystal Display Systems Ltd., [Mrežno]. Available: http://crystal-display.com/components/touchscreens/projective-capacitive/. [Pokušaj pristupa 16. 5. 2016].

[5] »Eizo Global,« [Mrežno]. Available: http://www.eizoglobal.com/library/basics/basic_understanding_of_touch_panel/. [Pokušaj pristupa 16. 5. 2016].

[6] »Impulse,« [Mrežno]. Available: http://www.impulse-corp.co.uk/touch-screen-lcd-displays-and-panel-pcs/touch-screen-comparison-table/infrared-touch-screen.htm. [Pokušaj pristupa 3. 7. 2016.].

[7] »libgdx github,« [Mrežno]. Available: https://github.com/libgdx/libgdx. [Pokušaj pristupa 22. 2. 2016.].

[8] »libgdx github,« [Mrežno]. Available: https://github.com/libgdx/libgdx/wiki/Event-handling. [Pokušaj pristupa 16. 3. 2016.].

[9] T. Cornez i R. Cornez, Android Programming Concepts, Burlington, MA: Jones & Bartlett Learning, 2015.

[10] Google, [Mrežno]. Available: https://source.android.com/devices/sensors/sensor-types.html. [Pokušaj pristupa 22. 3. 2016.].

[11] Google, [Mrežno]. Available: https://source.android.com/devices/input/touch-devices.html. [Pokušaj pristupa 17. 5. 2016].

[12] »racunalo.com,« [Mrežno]. Available: http://www.racunalo.com/kako-radi-kapacitivni-sustav-za-raspoznavanje-dodira-u-smartfonima-i-tabletima-ucimo-zajedno/. [Pokušaj pristupa 18. 5. 2016].

[13] »zeendo.com,« [Mrežno]. Available: http://zeendo.com/info/what-type-your-touch-scren-is/. [Pokušaj pristupa 18. 5. 2016.].

[14] S. Kolokowsky i T. Davis, »Touchscreens 101: Understanding Touchscreen Technology and Design,« p. 5, 2009.

[15] »Tutorial Libgdx Android,« [Mrežno]. Available: http://tutorial-libgdx-android.blogspot.hr/2014/02/inputs-handling-1st-part.html. [Pokušaj pristupa 2. 5. 2016.].

SAŽETAK

Usporedba načina upravljanja aplikacijom različitim senzorima

Rad je napisan s ciljem usporedbe načina kretanja lika na zaslonu korištenjem različitih senzora. U teorijskom dijelu rada objašnjene su prednosti senzora na Android uređajima, kao i generalna podjela senzora te vrste senzora podržane na Android operacijskom sustavu. Objašnjen je način prikupljanja podataka senzora i upotreba istih. Veća pozornost je posvećena samim senzorima koji su korišteni u stvaranju aplikacije, a to su akcelerometar i zaslon osjetljiv na dodir. Praktični dio rada je izveden u obliku Android aplikacije, koja omogućuje kretanje lika koristeći senzore na tri različita načina. U radu su prikazani dijelovi koda koji prikazuju implementaciju senzora u aplikaciji, kao i slike zaslona aplikacije. Na kraju je dan zaključak uz komentar samih rezultata i teme.

Ključne riječi: Android, aplikacija, senzori, akcelerometar, zaslon osjetljiv na dodir

ABSTRACT

Application control methods by using variuos sensors

Paper was written with the purpose of comparing ways of character's movement on the screen using different sensors. The theoretical part explains the advantages of sensors on Android devices, as well as the general division of sensors and sensor types that are supported on Android operating system. Also explained are ways of collecting data from sensors and their use. Big part of paper is dedicated to the very sensors that were used in the application and those are accelerometer and touch screen. Practical part of the work is carried out in the form of Android application which allows movement of the character using sensors in three different ways. The paper presents parts of code that show implementation of sensors in the application, as well as screenshots of application. At the end of the paper conclusion holds note of the results and the topic itself.

Key words: Android, application, sensors, accelerometer, touch screen

ŽIVOTOPIS

Marijeta Šaravanja, rođena 27. studenog 1991.godine u Đakovu, Republika Hrvatska.

Nakon završetka osnovne škole „Vladimir Nazor“ u Čepinu, upisala i završila Isusovačku klasičnu gimnaziju s pravom javnosti u Osijeku.

Po završetku iste, 2010.godine, upisala preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. Preddiplomski studij završila 2013.godine, te iste godine upisala diplomski studij procesnog računarstva.