SZÉCHENYI ISTVÁN EGYETEM
SZE - GYŐR
MŰSZAKI TUDOMÁNYI KAR

Informatika Tanszék

SZAKDOLGOZAT

Automatikus egyenestartást-segítő alkalmazás Androidra

Trencsik János

Mérnök Informatikus BSc szak

2014

Feladatkiírás

Szakdolgozat címe: Automatikus egyenestartást-segítő alkalmazás Androidra

Hallgató neve: Trencsik János

Szak: Mérnök informatikus

Képzési szint: BSc

Típus: nyilvános

A szakdolgozat feladat leírása:

Olyan Androidon futó alkalmazás, amely képes az okostelefon rezgetőjét és szenzorjait használni annak érdekében, hogy vakok vagy bárki más egyenesen tudjon haladni vizuális információ nélkül (pl. sötétben gyalogolva, referencia nélkül erdőben stb.).

- okostelefonok rezgetőjének és szenzorjainak tanulmányozása
- szenzorok adatainak kiolvasása
- helyes útvonalról való letérés kezelése
- az alkalmazás tesztelése

Győr, 2014. február 17.

_____	_____
belső konzulens	tanszékvezető
_____	
belső konzulens	

Nyilatkozat

Alulírott, Trencsik János (CQYOYB), mérnök informatikus, BSc szakos hallgató kijelentem, hogy az Automatikus egyenestartást-segítő alkalmazás Androidra című szakdolgozat feladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Győr, 2014. november 26. _____

hallgató

Kivonat

Automatikus egyenestartást-segítő alkalmazás Androidra

A szakdolgozat egy vakok egyenes vonalú haladását segítő Android alkalmazás fejlesztésének folyamatait mutatja be. A Straight Line Walk alkalmazás egy olyan megoldást biztosít, amellyel GPS segítsége nélkül tud a felhasználó egyenesen irányba haladni kültérben és beltérben egyaránt. Mindezt úgy, hogy a lehető legérthetőbb és legkönnyebb módon irányítsa a felhasználót rezgésekkel és/vagy hangokkal a helyes irány megtartása érdekében.

A dolgozat 1. fejezetében kerül sor a feladat részletes bemutatására. Ebben a fejezetben említésre kerülnek még vakoknak készült okostelefon alkalmazások és megoldások.

A 2. fejezetben az Android platform kerül áttekintésre, kiemelve az Android felépítését és főbb komponenseit. A fejezet végén bemutatásra kerülnek az okostelefonokban megtalálható szenzorok fajtái és elméleti működésük.

A dolgozat 3. fejezetében a Straight Line Walk alkalmazás tervezési folyamata következik. Use case diagramok segítségével specifikálásra kerülnek a rendszer funkciói, működése. Ebben a fejezetben történik az alkalmazás felhasználói felületének tervezése.

Az implementálási munka áttekintésére a 4. fejezetben kerül sor. Itt kerülnek taglalásra az egyes osztályok és főbb metódusaik. Ezek áttekintése során bemutatásra kerül az alkalmazás felhasználói felületének kialakítása is.

Az implementálás után a specifikáció szerinti működés igazolására kerül sor a teszteléssel az 5. fejezetben. Ebben a fejezetben végigvisszük az alkalmazást az egyes teszteseteken, amelyek főleg a felhasználói felületre vonatkoznak. Azonban az alkalmazás jellegéből adódóan a rendszer pontosságának tesztelésére is sor kerül.

A dolgozatból természetesen nem maradhat ki a felhasználói dokumentáció sem. A 6. fejezetben kerül bemutatásra a rendszer működése a felhasználó szempontjából. Minden egyes képernyő működése taglalásra kerül.

Az utolsó, 7. fejezet értékeli az elkészült feladatot és néhány továbbfejlesztési javaslattal zárul.

Abstract

Android application supporting straight line walk

The diploma thesis presents the development process of an Android-based application responsible for helping the visually impaired avoid veering while walking straight. The Straight Line Walk application is the result of work done. The application provides a solution, which helps for blind and visually impaired to guide the walk in straight line indoor and outdoor without a GPS. The goal of the application was to guide the user by the indications (vibration, sound, speech) of the smartphone.

The first chapter of the diploma thesis gives a detailed presentation of the task. In this chapter the solutions and techniques for smartphone applications are also mentioned.

The second chapter contains an overview about the Android platform, highlighting its architecture and main components. The end of the chapter gives a conspectus about the types of sensors which can be built in smartphones.

The third chapter continues with the design process of the Straight Line Walk application. The functions of the application are specified with Use case diagrams. The chapter describes the user interface of the application.

The design process is followed by the implementation part. The fourth chapter presents the classes and its methods, which are used in the source code. The chapter also presents the user interface and its functions via source code.

In the fifth chapter the operation of the application according to specification is verified by testing. Test cases are made for user interface, but according to the nature of the application, the accuracy of the application is also tested.

The user manual cannot be left out. The sixth chapter contains a user manual. Every single function is explained to the user.

The last chapter closes the diploma thesis with some proposals for enhancement.

Tartalomjegyzék

Bevezetés

Napjainkban a számítógépek használata szinte teljesen a mindennapok részévé vált. Ma már léteznek *vak és gyengénlátó* emberek számára készített különböző képernyőolvasó szoftverek, amik a képernyőn látható tartalmakat olvassák fel, és tájékoztatják a felhasználót a rendszerben történtekről. Ez a megoldás olyannyira hatékonynak bizonyult, hogy vannak látássérült emberek, akik számítógépen dolgoznak, sőt léteznek vak programozók is. [Pandur, 2012]

Manapság a legelterjedtebb kommunikációs eszköz az *okostelefon*. Sajnos a mai okostelefonok már csak minimálisan használnak fizikai gombokat. Legfeljebb a hangerő-szabályozó és képernyő kioldó gombja van a készüléken. Vakok és gyengénlátóknak első benyomásra használhatatlannak tűnnek ezek, mivel fizikailag nem érzékelhetők az érintőképernyőn megjelenített grafikai elemeket. Felmerül a kérdés, hogy vakok számára mennyire használható egy okostelefon. Látók számára rengeteg lehetőséget biztosít egy ilyen készülék, vakok számára azonban még a telefonálás is nagy feladat. Pedig ha jobban belegondolunk egy jól, vakok számára külön megtervezett alkalmazás nagy előny számukra. Léteznek különböző megoldások (pl. képernyőolvasók), amik hang vagy beszéd alapon próbálják tájékoztatni a felhasználót. Ezen az alapon tudunk segítséget nyújtani egy vak ember számára az okostelefonnal.

A szakdolgozat célja olyan módszer kutatása és megvalósítása, amely vakok és gyengénlátók *navigációját* segíti utcán vagy beltérben való haladáskor. Ehhez egy *Android alkalmazás* kerül fejlesztésre, aminek lényege, hogy GPS jel nélkül egyenes vonal mentén irányítja a felhasználót más egyéb *szenzorok* adatainak felhasználásával.

A feladat ötletét a Távközlési Tanszékről Dr. Wersényi György és a montréáli McGill Egyetem egy projektje szolgáltatta. A McGill Egyetem a projektet 2013-ban publikálta az International Conference on Auditory Display konferencián Lengyelországban [McGill, 2013][1]. A McGill Egyetem projektje iPhone készülékekre lett tervezve. Tudomásom szerint a projektet nem sikerült végigvinniük.

Felkeltette az érdeklődésemet ez a téma, mivel érdeklődök a mobiltechnológiák iránt és voltak már Android fejlesztéssel kapcsolatos tapasztalataim. Ezenkívül szerettem volna mélyebb betekintést nyerni az okostelefonok szenzorjainak működésébe. Úgy érzem ezzel a feladattal a vakok és gyengénlátók életét, ha egy kicsit is, de megkönnyíthetem.

1. Automatikus egyenestartást-segítő alkalmazás

1.1. A feladat ismertetése és az alkalmazás célja

A szakdolgozatom célja egy olyan alkalmazás fejlesztése, amely vakok egyenes vonalú haladását segíti. Az *egyenestartást segítő alkalmazás* az okostelefonok szenzorjai segítségével fog működni. A felhasználó a kezében tartva a telefont az általa választott irányba áll és a képernyő érintésével indíthatja az egyenes haladást. Ha a felhasználó haladás közben kitér az egyenes irányból, akkor az alkalmazás rezgésekkel, ill. hangvezérléssel tájékoztat és ösztönöz a megadott egyenes haladás betartására. Ha a felhasználó jobbra, vagy balra kíván 90 fokos fordulatot tenni akkor a képernyő jobb vagy bal oldalának érintésével megteheti. Természetesen a gyengén látók számára *akadálymentesített felületet* is tartalmaz. Az alkalmazás indulásakor a felhasználó a képernyő érintésével tájékoztatást kap hangfelvételen keresztül az alkalmazás működéséről.

Az alkalmazás platformjául az `Android`-ot választottam, mivel ez a legelterjedtebb a felhasználók körében és az alacsonyabb árú készülékeken is ez fut. Emellett rugalmas és biztonságos fejlesztést tesz lehetővé, rengeteg felhasználó érhető el vele. Mivel az Android készülékek több, mint 85%-án a 4.0-s vagy újabb Android verzió található [2], ezért az alkalmazásom is ezekre a verziókra lesz optimalizálva.

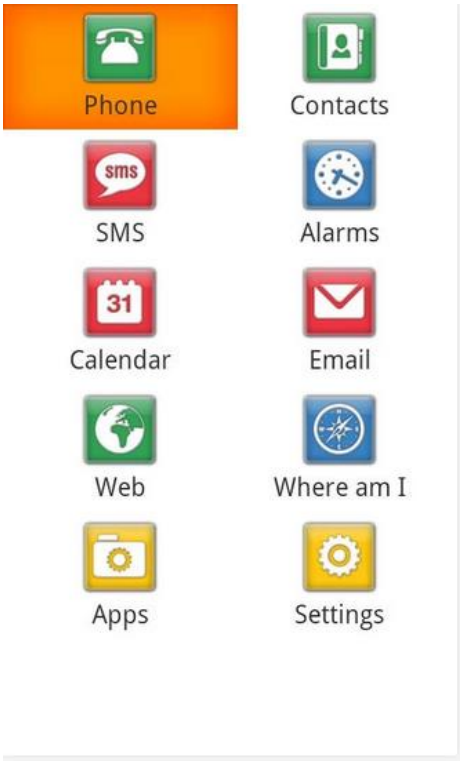
1.2. Vakok és gyengénlátóknak szánt létező megoldások bemutatása

Érintőképernyős okostelefonok általában nehézséget okoznak egy vak vagy gyengénlátó számára. Problémát jelent, hogy a képernyőn a különböző összetett felületi komponensek a képernyő-felolvasó segítségével nehezen kezelhetők. Egy vak ember számára az a legfontosabb, hogy minél egyszerűbben és könnyen kezelhetően legyenek elrendezve a képernyőn az egyes felületi komponensek. Természetesen rendelkezésre áll többféle megoldás, ám ezek az alkalmazások gyakran csak részleges megoldással szolgálnak.

A `Slide Rule` olyan megoldás, amelybe a legfontosabb alkalmazásokat építették bele, mint például a telefonkönyv, üzenetküldés és a zenelejátszás funkciók. A fejlesztők

nem a grafikus felhasználói felületre koncentráltak, hanem az egyszerű kezelhetőségre és a beszéd alapú visszacsatolás minél hatékonyabb tervezésére. Ez a rendszer egy listát használ, amin a fel-le gesztusok segítségével tud a felhasználó lépkedni, illetve a jobbra-balra gesztusokkal érhetőek el az egyes listaelemek funkciói [Pandur, 2012]. Például egy SMS küldésénél, ha a felhasználó a küldés funkciót szeretné elérni, akkor jobbrahúzás gesztussal ezt meg tudja tenni, ahelyett, hogy a küldés gombot keresgélné.

A `Mobile Accessibility` egy olyan alkalmazás, amely több hozzáférhető funkcióval rendelkezik (pl. telefon, névjegyzék, SMS, ébresztő, naptár, e-mail, web stb.), amik vakok és gyengénlátóknak lettek tervezve. Mindegyik funkciónak egy leegyszerűsített felhasználói felülete van, aminek a szöveges információit hangalapú, verbális visszacsatolással közli a rendszer a felhasználó felé. Az alkalmazás egyben képernyő-felolvasó funkcióval is rendelkezik, aminek segítségével operációs rendszer szinten a standard felhasználói felületen is verbális információk segítségével tud navigálni a felhasználó. A Mobile Accessibility rendszer egy listaalapú felülettel rendelkezik **[1. ábra].** A listaelemek egyszeri érintésével a rendszer szövegesen közli a felhasználóval, hogy melyik elemen tartózkodik éppen. Duplaérintéssel pedig az adott listaelem funkciói érhetők el. Az alkalmazás tartalmaz pozíció meghatározó funkciót („Where am I”), amely a GPS alapján visszaadja a készülék pozícióját. Természetesen a GPS miatt ez csak kültérben használható jól. Az alkalmazás rendelkezik néhány hátránnyal is. A menüpontok közötti navigáció lassú és hiányoznak funkciók néhány helyen. Ezenkívül a teljes verziós alkalmazás ára nagyon magas [3]. Az alkalmazás előnye, hogy több nyelven is elérhető, ám minden nyelven külön le kell tölteni az alkalmazást.



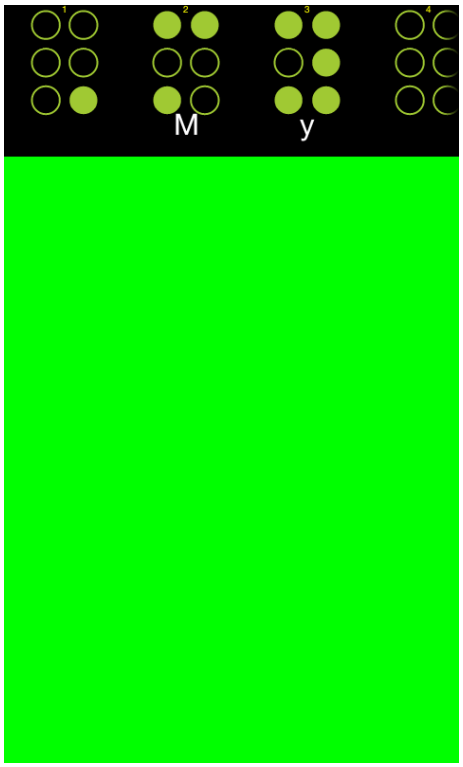
1. ábra. Mobile Accessibility

A `Georgie` ingyenesen letölthető alkalmazás operációs rendszer szinten épül be, és nyújt a vakok és gyengénlátók számára képernyőfelolvasó funkciót. A felhasználói felület itt is listaalapú, ám csak egy oszlopa van, így a Mobile Accessibility alkalmazáshoz képest a felületi elemek nagyobbak és egyszerűbben kezelhetőek a felhasználó számára. Úgy működik, hogy egy felhasználói felületi vezérlő komponensre egyszer röviden nyomva a rendszer verbálisan közli az információt a komponensről. A komponensbe lépni pedig hosszan nyomva lehet. Sajnos léteznek olyan vezérlő komponensek, amiknél egyáltalán nem használható (főleg letöltött alkalmazásoknál). A Georgie ingyenes verziója bővíthető olyan funkciókkal, mint a telefonálás, emlékeztető, Youtube videó lejátszó, hangfelvevő, szövegfordító, időjárás, kamera, webböngésző, OCR alapú szövegfelismerő, kereső, közeli helyek és a navigáció. Utóbbi hátránya, hogy ehhez is GPS szükséges. Ezen bővítmények ára sajnos elég borsos lehet a felhasználóknak [4].

A szövegbeviteli lehetőségek vakok és gyengénlátóknak szintén elég korlátozottak. A szövegbevitel megoldható a virtuális billentyűzet és képernyőfelolvasó együttes használata segítségével. Ellenben, ha például törölnie kell a felhasználónak, akkor eléggé lassúvá és körülményessé teszi egy szöveg megírását. Ennek kiküszöbölésére használhatóak

beszédfelismerő rendszerek. Nagyon sok androidos készülék tartalmaz alapértelmezetten beszédfelismerő funkciót is. Ezenkívül az *alkalmazásboltból* is letölthető többféle. A hasonló mai rendszerek hátránya, hogy a beszédminták dekódolására egy távoli adatbázist használnak, ezért a használatukhoz internetkapcsolat szükséges [Pandur, 2012]. Emellett a hosszabb mondatokat nem képesek pontosan felismerni. A beszédfelismerő rendszerek azonban jól használhatóak és meggyorsítják a szövegbevitelt is. Manapság már magyar nyelvű beszédfelismerő is létezik.

A szövegbevitel megkönnyítésére már léteznek olyan alkalmazások, amik a Braille írást használják. A `SmartBraille` alkalmazás ezt a módszert használja szövegek bevitelére **[2. ábra].**



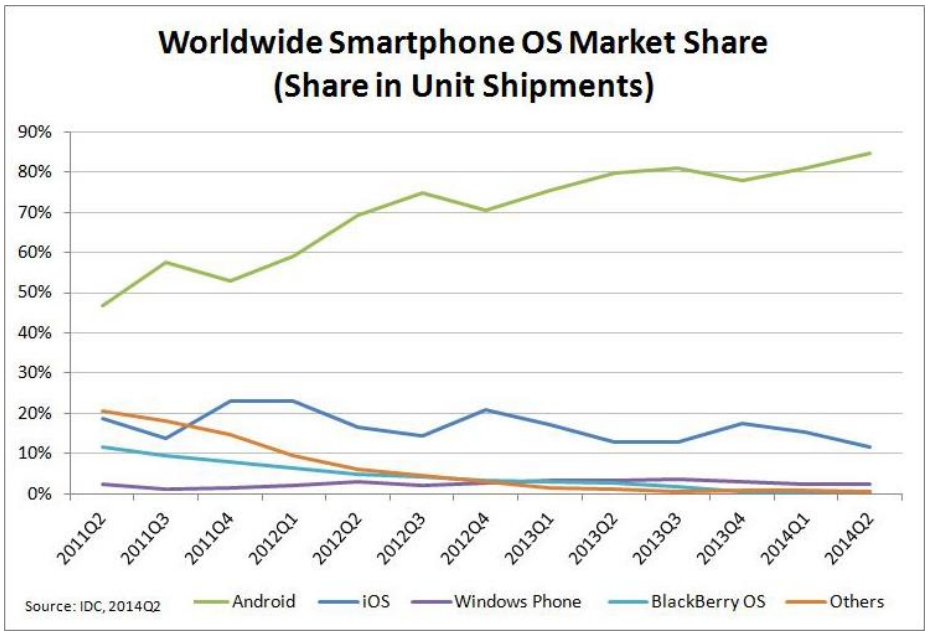
2. ábra. SmartBraille

E szakdolgozat témája azért is érdekes, mert olyan jellegű alkalmazás, ami GPS nélkül irányítaná a vak vagy gyengénlátó felhasználót, nincs az alkalmazásboltokban. Az elérhető hasonló alkalmazások mind GPS-t használnak alapul a navigációhoz (pl. Georgie), ezért csak kültérben használhatóak igazán.

2. Android

Az Android operációs rendszer jelenleg a legnépszerűbb mobil készülékeken futó szoftver platform [5]. A Google által fejlesztett rendszer egy `Linux` kernel magot használ alapul. Ez a rendszermag biztosítja egyedülálló módon a rendszer hordozhatóságát. Azaz lényegében bármely hardver platformon képes futni. Az Androidot open-source projektként fejlesztik az Apache 2.0 és a GPLv2 licenc alatt. Lényegében ez biztosítja, hogy a rendszer ingyenes legyen. A fejlesztők semmilyen módon nincsenek megkötve, ezért szabadon tudnak készíteni bármilyen felhasználói felületet vagy felhasználhatják a készülék minden elérhető funkcióját.

Az Android egy nyílt forráskódú operációs rendszer, amelynek a kereslete egyre növekszik a piacon. Ez azt eredményezi, hogy egyre több alkalmazást fejlesztenek erre a platformra. A legnagyobb konkurensei az `Apple iOS` és a `Microsoft Windows Phone`. Ezek legfőbb hátránya a különböző megkötések és szigorú licenc feltételek a fejlesztőkkel szemben.



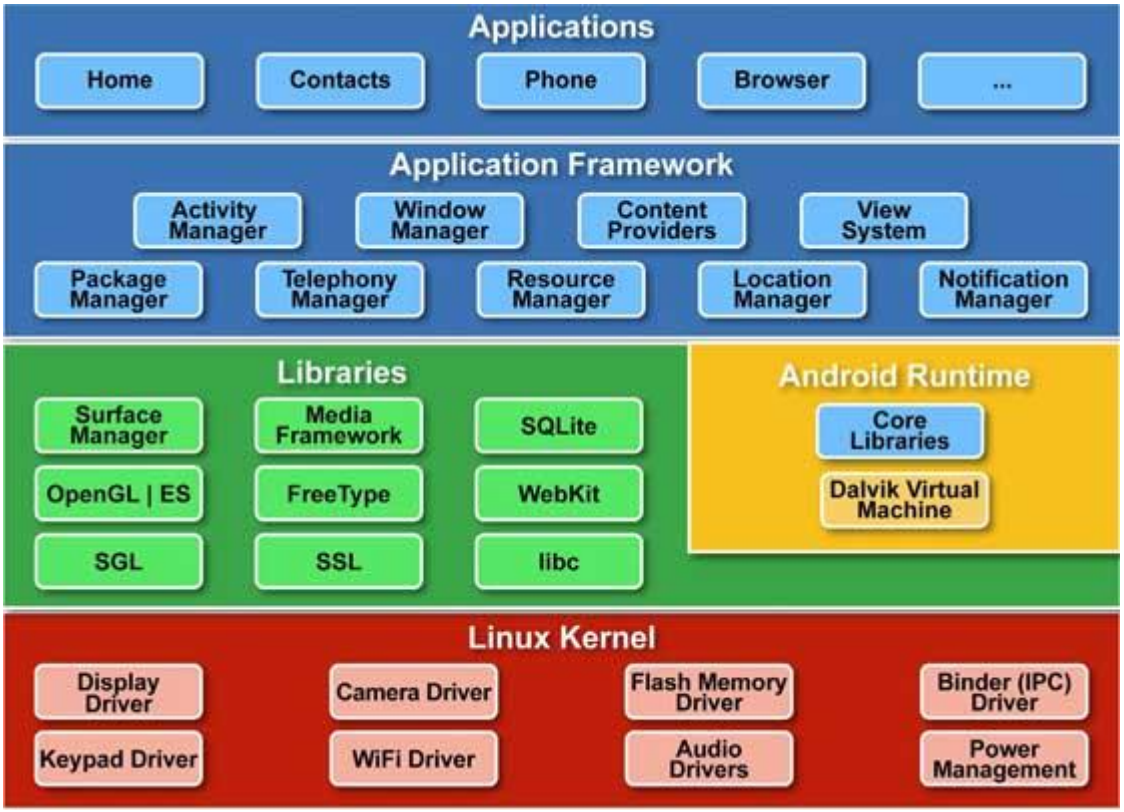
3. ábra. Okostelefon operációs rendszerek piaci eloszlása, 2014 Q2 [6]

2.1. Története

Az Android Inc. vállalatot Andy Rubinem és társai alapították az USA-ban 2003-ban. Mobil alkalmazások fejlesztésével foglalkoztak. Nagyon hamar pénzügyi gondjaik támadtak. A Google megmentette a vállalatot azzal, hogy 2005-ben felvásárolta leányvállalataként. Az eredetileg az Android Inc. által fejlesztett Android platformot a Google fejlesztette tovább. 2007-ben megalapították az Open Handset Alliance (OHA) csoportot, amely hivatalosan bemutatta az új open-source Android szoftvert. Az OHA tagjai mobilhálózat üzemeltetők, mobil készülékek gyártói, alkalmazásfejlesztők és kereskedelmi cégek [7]. Az első forgalmazott androidos mobil készülék 2008 őszén került a piacra az USA-ban.

2.2. Architektúra

Az Android platform architektúrája 5 rétegen alapszik. Minden rétegnek meghatározott feladata van, de a rétegek közti határok el vannak mosódva **[4. ábra][**8].



4. ábra. Android rendszer struktúrája [8]

A legalacsonyabb réteg a Linux Kernel. Összeköttetést biztosít a hardver és az operációs rendszer között. Olyan alap rendszer funkcionalitásokat nyújt, mint folyamat menedzsment, memória menedzsment, eszköz menedzsment (pl. kamera, kijelző, billentyűzet). Emellett biztosítja a hálózatkezelést és az eszköz meghajtókat (device driver) [8].

Az architektúra következő részei különböző rendszerkomponensek által használt C/C++ könyvtárak. Ezek tartalmazzák például az open-source web böngésző WebKit motort, SQLite adatbázist, SSL könyvtárakat, médialejátszó keretrendszert stb.

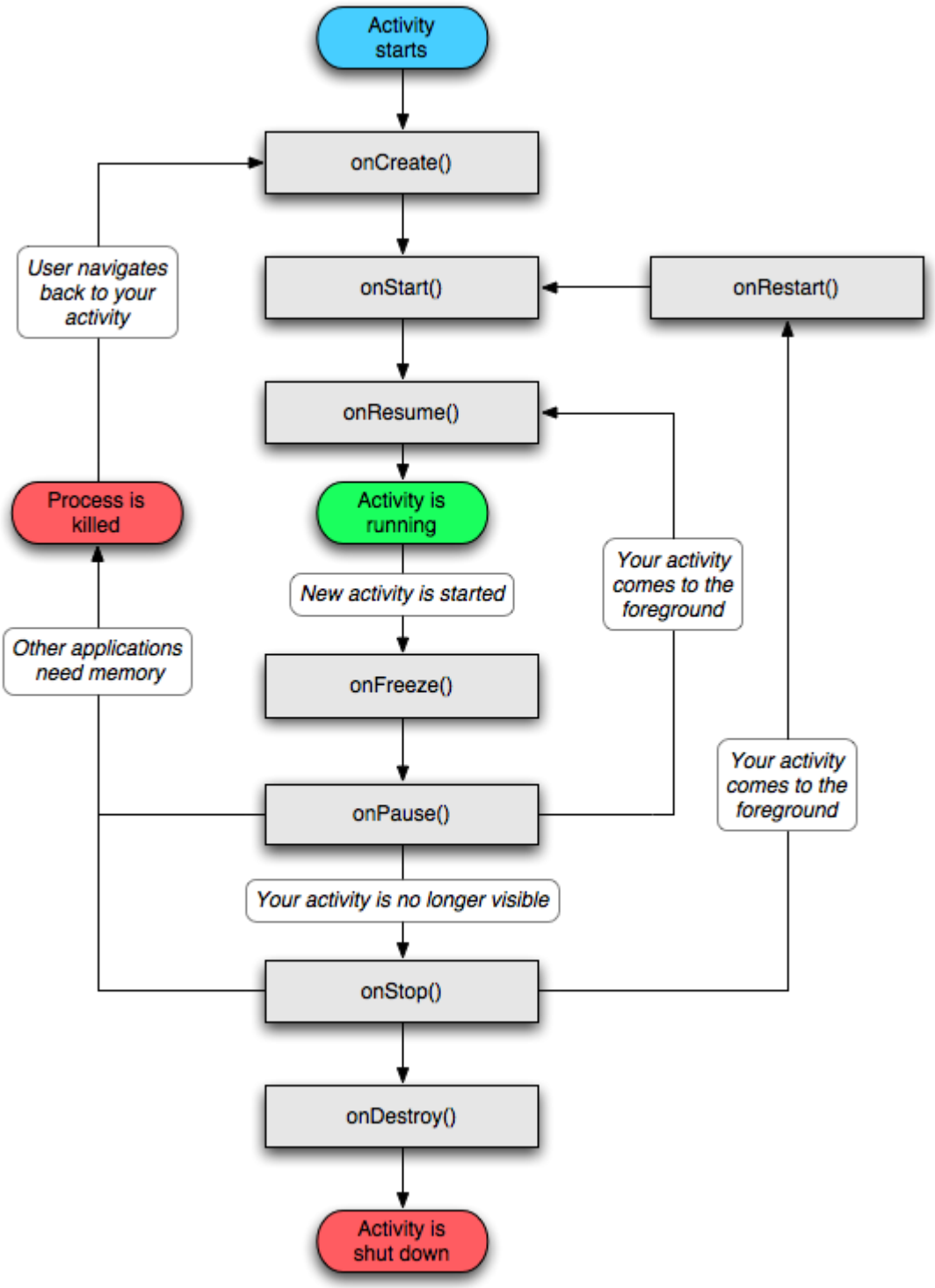
E felett található az Android futási környezet, ami tartalmazza a `Dalvik Virtuális Gépet` (Dalvik Virtual Machine) és olyan könyvtárakat, amelyek ahhoz szükségesek, hogy a fejlesztők a standard Java programozási nyelven tudjanak alkalmazást fejleszteni. A Dalvik Virtuális Gép tulajdonképpen egy olyan `Java Virtuális Gép` (Java Virtual Machine - JVM) amely kifejezetten az Android részére lett tervezve és optimalizálva. Feladata az Androidra készített Java alkalmazások futtatása. Az Android alkalmazások futtatását biztonságossá teszi, mert így csak kicsi eséllyel tudja megbénítani az egész rendszert egy alkalmazás [8] [Ekler et al., 2012].

A legfelső rétegben az Alkalmazás Keretrendszer Java osztályokat tartalmaz, amelyek magas szintű szolgáltatásokat biztosítanak az alkalmazásoknak. A fejlesztőknek megengedett, hogy használják ezeket a szolgáltatásokat alkalmazásuk fejlesztése során. Minden alap Android alkalmazás az Alkalmazás rétegben van telepítve. Az alkalmazások két kategóriába sorolhatók. Az első kategória a natív alkalmazások, melyek előre telepített programok, a rendszerrel való munkát szolgálják, mint például naptár, kapcsolatok, térképek, stb. A második kategória a kereskedelmi alkalmazások, amelyek harmadik fél típusú programok, mint például ezen szakdolgozatban fejlesztett alkalmazás is, amelyet a felhasználó telepít fel. Android platformra .apk állományokat telepíthetünk, amelyek becsomagolva tartalmazzák magát az Android alkalmazást. A telepítést az úgynevezett PackageManagerService végzi.

2.3. Alkalmazás komponensek

Az Android alkalmazások több különböző komponensből épülhetnek fel, amelyek most bemutatásra kerülnek.

Az *Activity* egy felhasználói felülettel rendelkező képernyőt reprezentál. Például egy e-mail alkalmazásnak lehet egy Activityje, ami egy listában megjeleníti az új leveleket, és egy másik Activity, ami az adott levél olvasására alkalmas. Ha egy alkalmazás több Activityből áll, mindig van egy kezdő, fő Activity, amely az alkalmazás indulásakor jelenik meg. Bármelyik Activityből indulhat újabb. Ha egy új Activityt elindítunk, az előző Activity leáll (Stopped), de a rendszer megőrzi a Back Stack-en. Amikor az új Activity elindul, rákerül a Back Stack-re, és megkapja a vezérlést (focus). A Back Stack lényegében egy LIFO veremként (Last In First Out – az utoljára eltárolt elem kerül ki elsőnek a veremből) szolgál, amely a példányosított Activityket tartalmazza, de mindig csak egy van aktív állapotban. A Vissza (Back) gomb megnyomásakor a stacken legfelül lévő Activity kikerül a veremből, és az alatta levő lesz aktív [Ekler et al., 2012]. Amikor egy Activity leáll egy másik indulása miatt, mindkét Activity-oldal értesítést kap az eseményről az úgynevezett életciklus-metódusokon keresztül **[5. ábra].** Az Activity életciklusában három állapotba kerülhet. Amikor az Activity előtérben van, felhasználói inputokat fogad, futó (running/resumed) állapotban van. Abban az esetben, ha egy másik Activityvel van elfedve, úgynevezett ideiglenes elfedésben van és nem fogad felhasználói inputokat, akkor szüneteltetett (paused) állapotban van. Az utolsó állapot, amikor az Activity már egyáltalán nem látható, de az objektum a memóriából még nincs törölve, akkor leállított (stopped) állapotban van. Amikor egy Activity állapotot vált, a megfelelő callback metódusokat a rendszer meghívja. Viszont e metódusok helyes implementációja a fejlesztők feladata.



5. ábra. Activity életciklusa [9]

Callback metódusok általános feladatai a következők:

- `onCreate()`: Meghívásra kerül, amikor az Activity először létrejön és beállítja a megfelelő állapotokat.
- `onDestroy()`:Meghívásra kerül, amikor az Activityt megsemmisíti a rendszer. Minden még foglalt erőforrás felszabadítása megtörténik.
- `onStart()`: Az Activity láthatóvá válásakor hívódik meg. A rajta lévő vezérlők kezelhetők lesznek. Ilyenkor szokás pl. a Broadcast Receiverekre feliratkozni (lásd később).
- `onStop()`: Az Activity már nem látható. Ilyenkor érdemes leiratkozni a Broadcast Receiverekről.
- `onRestart()`: Az Activity leállítása utáni újraindításakor kerül meghívásra.
- `onResume()`:Meghívásra kerül, ha a felhasználó interakcióba lép az alkalmazással. Az Activity az előtérbe kerül, láthatóvá válik, a felhasználó számára kezelhetővé válnak a vezérlők.
- `onPause()`: Meghívásra kerül, ha a másik Activity kerül fókuszba, de az eredeti Activity valamennyire még látszik a háttérben (pl. dialógus ablak felugrásakor).

Az Activityvel ellentétben a *Service (szolgáltatás)* nem rendelkezik felhasználói felülettel, csak egy hosszabb ideig a háttérben futó folyamatot reprezentál. Ugyanúgy indíthat új Activityt vagy valamilyen felugró ablakot. A szolgáltatásokkal általában olyan funkciókat szokás megvalósítani, mint például zenelejátszás, letöltések, helymeghatározás stb. A szolgáltatásokat alapvetően kétféle módon érhetjük el. A Started Service-nek a végrehajtandó parancsot Intent formájában adjuk át. A szolgáltatás megkapja a parancsot, majd elvégezi a műveletet. A hívó komponens értesítéséről külön kell gondoskodni. A Bound Service-nél a szolgáltatás egy saját, a programozó által definiált interfészt bocsát a hívó fél rendelkezésére. Ellentétben a Started Service-ek Intent-alapú parancsaival, a Bound Service tetsző1eges számú és típusú metódust definiálhat a szolgáltatással való kommunikációhoz. Szolgáltatásokat igénybevevő alkalmazások fejlesztésénél ügyelni kell arra, hogy feleslegesen ne fusson a háttérben nagy adatforgalmat generálva, ami a készülék akkumulátorának lemerítésével járhat. [Ekler et al., 2012]

A *Content Provider* lényegében egy tartalomszolgáltató komponens. Feladata egy adatforrás kezelése és az adatforrásra vonatkozó kérések kiszolgálása. Az alkalmazások többféle módon tárolhatnak adatokat – fájlban, SQLite adatbázisban vagy az interneten –, éppen ezért ezekhez az adatokhoz más alkalmazásoknak is lehet hozzáférhetősége. Az adatokat nem csak alkalmazások, de az Activityk között is meg lehet osztani.

A *Broadcast Receiver* feladata, hogy reagáljon bizonyos események hatására és valamilyen feladatot hajtson végre. Az Android rendszer számos eseményt jelez broadcastok (sugárzás) formájában, amelyre feliratkozhatunk a saját alkalmazásunkból. Ugyanúgy, mint a szolgáltatás és a Content Provider esetén, a Broadcast Receiver sem rendelkezik felhasználói felülettel.

Minden ilyen építőelemet a gyökérkönyvtárban megtalálható AndroidManifest.xml fájlban kell definiálni. Ezt a fájlt minden Android alkalmazásnak tartalmaznia kell. Ez a fájl tartalmazza még a jogosultságokat és követelményeket a hardver és szoftver tulajdonságokra és a készülék konfigurációjára vonatkozóan.

A következő Android környezetben használt eszköz az *Intent*. Az Intent egy aszinkron üzenet. Az alkalmazáskomponensek adatcseréjének elsődleges módszere. [Ekler et al., 2012] Szolgálatot tesz még Activity, service és Broadcast Receiver indításánál.

Minden alkalmazásban definiálva vannak olyan elemek, mint például az ún. layout, kép, animáció, lokalizációs stringek. Ezeket erőforrásoknak (resource) nevezik. Az erőforrások optimalizálhatók különböző eszközökre, például a képernyő méretétől függően. Az erőforrások azonosítójuk alapján érhetőek el. Ezek az azonosítók az R.java állományban tárolódnak.

2.4. Android verziók

Újabb Android verziókat általában olyan hibák javítása céljából adnak ki, amik az előző verziókban fellelhetők voltak. Ezenkívül tartalmazhatnak még készüléktől függően új funkciókat is. A verziószámozás a változtatások méretétől függ. Abban az esetben, ha csak kisebb változtatás történt, akkor csak egy tizeddel növelik a verziószámot. Minden verziót egy desszertről neveznek el (lásd 1. táblázat).

1. táblázat. Android verziók [10]

Verzió	Kódnév	Kiadás dátuma	API szint	Piaci eloszlás
4.4	KitKat	2013. október 31.	19	30.2%
4.3.x	Jelly Bean	2013. július 24.	18	7.3%
4.2.x		2012. november 13.	17	20.8%
4.1.x		2012. július 9.	16	22.8%
4.0.3–4.0.4	Ice Cream Sandwich	2011. december 16.	15	8.5%
2.3.3–2.3.7	Gingerbread	2011. február 9.	10	9.8%
2.2	Froyo	2010. május 10.	8	0.6%

2.5. Szenzorok

Mivel a szakdolgozat témája erősen kötődik a szenzorokhoz, ezért a továbbiakban bemutatásra kerülnek az okostelefonok szenzorjai. A szakdolgozatomban az egyenestartást segítő alkalmazás fejlesztéséhez a készülék gyorsulásmérő és a mágneses mező szenzorjának értékei szolgálnak alapul.

Az Android operációs rendszert futtató eszközök rendelkeznek beépített szenzorokkal, amelyek képesek mérni a mozgást, orientációt és különféle környezeti feltételeket. Ezek a szenzorok nagy pontossággal képesek adatokat szolgáltatni. Lehetőség van például az eszköz mozgásának vagy pozíciójának megfigyelésére mindhárom dimenzióban, vagy ha a környezeti változásokat akarjuk figyelni az eszköz közelében. A mozgásérzékelő szenzorok leggyakrabban játék alkalmazásokban használatosak forgatás,

rázás, billenés stb. érzékelésére.

Az Android platform 3 általános szenzortípust biztosít:

- Mozgásérzékelő szenzorok:

  Mérik a gyorsulási és a rotációs erőket a 3 tengely mentén. Ide sorolható a gyorsulásmérő (accelerometer), gravitációs szenzor (gravity sensor), giroszkóp (gyroscope) és a rotációs vektor (rotational vector) szenzor.

- Pozíciót meghatározó szenzorok:

  Az eszköz fizikai pozícióját mérik. Ebbe a kategóriába tartozik az orientáció (orientation sensor) és a mágneses mező szenzor (magnetic field sensor).

- Környezeti szenzorok:

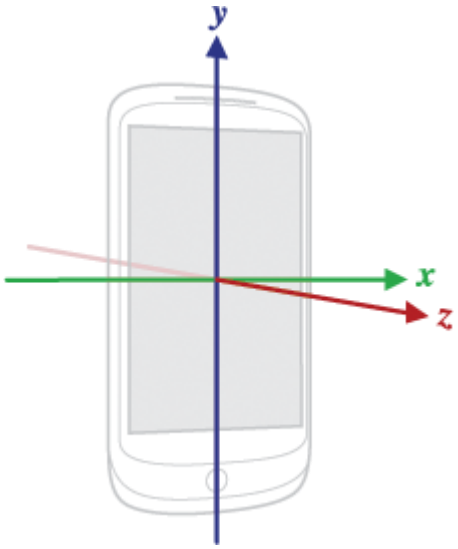
  Olyan környezeti paramétereket mérnek, mint a környező levegő hőmérséklete, légnyomás, fényerősség és a páratartalom. Ide sorolható a barométer, fotométer és a hőmérő.

Az Android szenzor keretrendszere (sensor framework) hozzáférést nyújt több típusú szenzorhoz. Létezik hardver és szoftver alapú szenzor is. A hardver alapú szenzorok fizikailag beépített komponensei a készüléknek. Az adataikat egyenesen a környezettel kapcsolatos mérni kívánt tulajdonságokból szerzik, mint például gyorsulás, mágneses mező erőssége vagy szögsebesség változás. A szoftver alapú szenzorok nem fizikai berendezések, habár úgy viselkednek, mintha hardver alapúak lennének. A szoftver alapú szenzorok az adataikat egy vagy több hardver alapú szenzortól szerzik. Általában virtuális szenzornak nevezik ezeket. Az egyenesvonalú gyorsulás (linear acceleration) szenzor és a gravitációs szenzor jó példák szoftver alapú szenzorokra. Android platformon a szenzortípusok konstansként vannak definiálva (pl. TYPE_GYROSCOPE, amelynek értéke 4) és ezekkel hívhatók meg.

A drágább Android készülékekben (pl. Samsung Galaxy S5) megtalálható minden típusú, mobileszközökben alkalmazható szenzor, azonban a legtöbb olcsóbb készülékek (pl. HTC Desire 200) csak néhánnyal rendelkezik. Természetesen az adott fajtájú szenzorból lehet akár több is egy készülékben, például egy eszköz tartalmazhat két gravitációs szenzort, de mindegyik más tartományban mér.

A szenzor keretrendszer a standard 3-tengelyű koordináta rendszert használja viszonyítási alapként. A szenzorok többsége az **6. ábrán** látható módon van

alapértelmezetten tájolva az eszközhöz viszonyítva. Amikor a készülék az alapértelmezett irányban helyezkedik el, akkor az X tengely horizontális és jobbra irányul, az Y tengely vertikális és felfelé irányul, a Z tengely pedig merőlegesen áthalad a képernyőn. Ezt a koordinátarendszert használja a gyorsulásmérő, gravitációs szenzor, giroszkóp, egyenes vonalú gyorsulás szenzor és a mágneses mező szenzor.



6. ábra. Android Sensor API által használt koordinátarendszer(az eszközhöz viszonyítva) [11]

A következő példakód mutatja, hogyan kapjuk meg a szenzor egy példányát (instanciáját):

```
private SensorManager mSensorManager;

private Sensor mSensor;

...

mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);

mSensor =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

2.5.1. Mozgásérzékelő szenzorok

A mozgásérzékelő szenzorok közül a gyorsulásmérő és a giroszkóp hardver alapú, míg a többi lehet hardver vagy szoftver alapú is egyaránt (pl. gravitációs, egyen vonalú gyorsulás és a rotációs vektor szenzorok). A mozgásérzékelő szenzorok az eszköz

mozgásának megfigyeléséhez használatosak. A mozgás lehet az eszköz megdöntése, megrázása, elforgatása vagy lengése. A mozgásérzékelő szenzorok önmagukban nem az eszköz pozíciójának meghatározására szolgálnak általában, de más szenzorokkal együtt használva, mint például a mágneses mező szenzor, már használhatóak az eszköz Földhöz viszonyított pozíciójának meghatározására. A SensorEvent osztály egy szenzor eseményt reprezentál, amely információkat hordoz. Ilyen információ lehet a szenzor típusa, időbélyeg, pontosság és természetesen a szenzor által mért értékek. A mozgásérzékelő szenzorok minden SensorEvent eseményre egy vektort adnak visszatérési értékül.

Gyorsulásmérő (Accelerometer):

Méri az eszközre alkalmazott gyorsulást, figyelembe véve a gravitációs erőt. SensorEvent esemény bekövetkezésekor visszatérési értékül három elemű tömböt ad, melynek első értéke az X tengelyre, második értéke az Y tengelyre, harmadik értéke a Z tengelyre mért gyorsulási erő m/s^2-ben megadva. Ezt a szenzort a szakdolgozatomban azért használom, hogy a mágneses szenzorral együtt meghatározható legyen a rotációs mátrix, amelyből egy metódushívás segítségével a készülék orientációja már könnyen visszakapható.

Giroszkóp (Gyroscope):

Az eszköz forgásának mértékét adja rad/s-ban az eszköz X, Y és Z tengelyére vonatkozóan. A szakdolgozat tervezésénél ez a szenzor is szóba került, mint lehetőség. Ha a készülék nincs mozgatásban, akkor az alapérték mindhárom tengelyre zérus. Csak a készülék mozgatása közben térnek el zérustól az értékek. Ha a felhasználó egyenesen halad, akkor megközelítőleg szintén zérus mindhárom tengelyen az érték, ezért a feladat szempontjából ennek a szenzornak a további használata elvetésre került.

Gravitációs szenzor (Gravity sensor):

Három elemű vektort ad vissza, jelezve a gravitáció irányát és mértékét m/s^2-ban.

2.5.2. Pozíciót meghatározó szenzorok

Az Android platform az eszköz pozíciójának meghatározásához a mágneses mező szenzort nyújtja. Ezen kívül a közelség szenzor méri, hogy milyen távol van a készüléktől egy tárgy. A mágneses mező szenzor és a közelség szenzor hardver alapú szenzorok. A készülékek többségében van mágneses mező szenzor. Ugyanúgy a közelség szenzor is gyakori a készülékekben, mert meghatározható vele, hogy a készülék a felhasználó arcához

van tartva (pl. telefonálás közben).

A pozíciót meghatározó szenzorok az eszköz Földhöz viszonyított fizikai pozíciójának meghatározására szolgálnak. Például a mágneses mező szenzor és a gyorsulásmérő kombinációjával meg lehet határozni az eszköz pozícióját a mágneses északi pólushoz viszonyítva.

Mágneses mező szenzor (Geomagnetic field sensor):

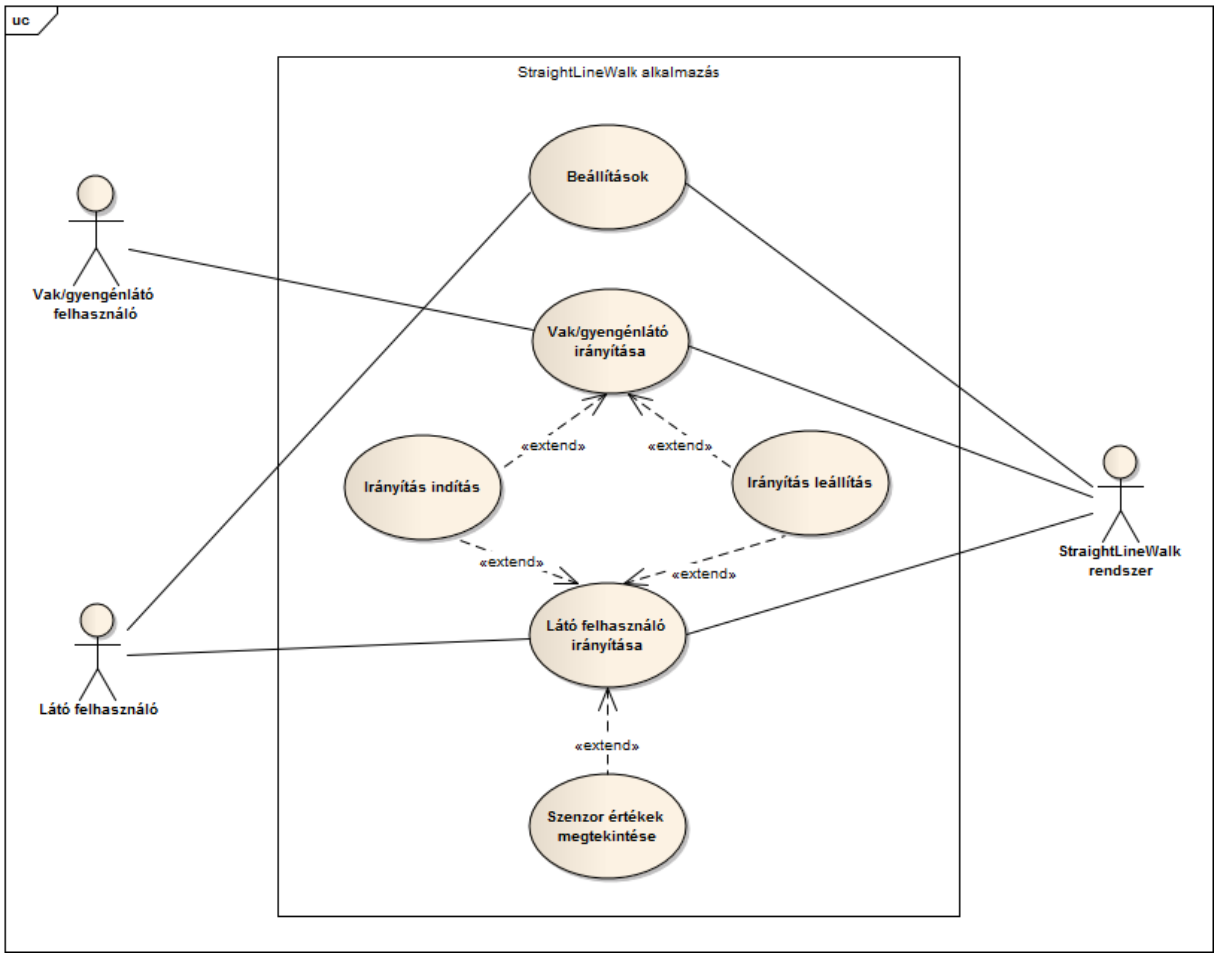
A Föld mágneses mezőjének változásait érzékeli. Ez a szenzor nyers μT-ban mért mágneses erősség adatokat ad vissza mindhárom koordinátatengelyre. A készülékhez közel lévő mágneses elemek befolyásolhatják a szenzort. Ez a szenzor rendkívül zajos is lehet, és ekkor lehet, hogy nem lehet korrektül kalibrálni. Ezek az események ahhoz vezetnek, hogy helytelen szenzor adatokat ad vissza. Ahogy már a gyorsulásmérőnél volt róla szó, ezt a szenzort is használom a szakdolgozatban a készülék orientációjának meghatározásához a gyorsulásmérővel együtt.

3. Tervezés

Az alkalmazásnak a `Straight Line Walk` nevet adtam. Ez jól kifejezi a lényeget és marketing szempontból is előnyös, mert könnyen megjegyezhető.

3.1. Use Case forgatókönyvek

Az alkalmazás funkciói áttekinthetők az **7. ábra** alapján.



7. ábra. Az alkalmazásunk által támogatott use case

Vak/gyengénlátó irányítása

Szereplők:

- Vak/gyengénlátó
- Straight Line Walk rendszer

Előfeltételek:

- A felhasználó a vakoknak és gyengénlátóknak szánt felületen van.

Lefutás:

1. A felhasználó a kezdőképernyőn (vakoknak és gyengénlátóknak szánt felület) az ujjával fel vagy le irányban végigsimítja a képernyőt.
2. A rendszer a telefon rezgetésével és/vagy hangvezérléssel irányítja a felhasználót az egyenes vonalú haladásban.

Látó felhasználó irányítása

Szereplők:

- Látó felhasználó
- Straight Line Walk rendszer

Előfeltételek:

- A felhasználó a kezdőképernyőről a látóknak szánt felületre navigál.

Lefutás:

1. A felhasználó a látóknak szánt felületen megnyomja a Start gombot. Eközben láthatóak a szenzorok adatai.
2. A kiválasztott irány értéke kikerül a képernyőre.
3. A rendszer a telefon rezgetésével és/vagy hangvezérléssel irányítja a felhasználót az egyenes vonalú haladásban.

Beállítások

Szereplők:

- Látó felhasználó
- Straight Line Walk rendszer

Következmények:

- A felhasználó által választott értékeket beállítja a rendszer az adott eszközön.
- A felhasználó által végzett beállításokat elmenti a rendszer.

Lefutás:

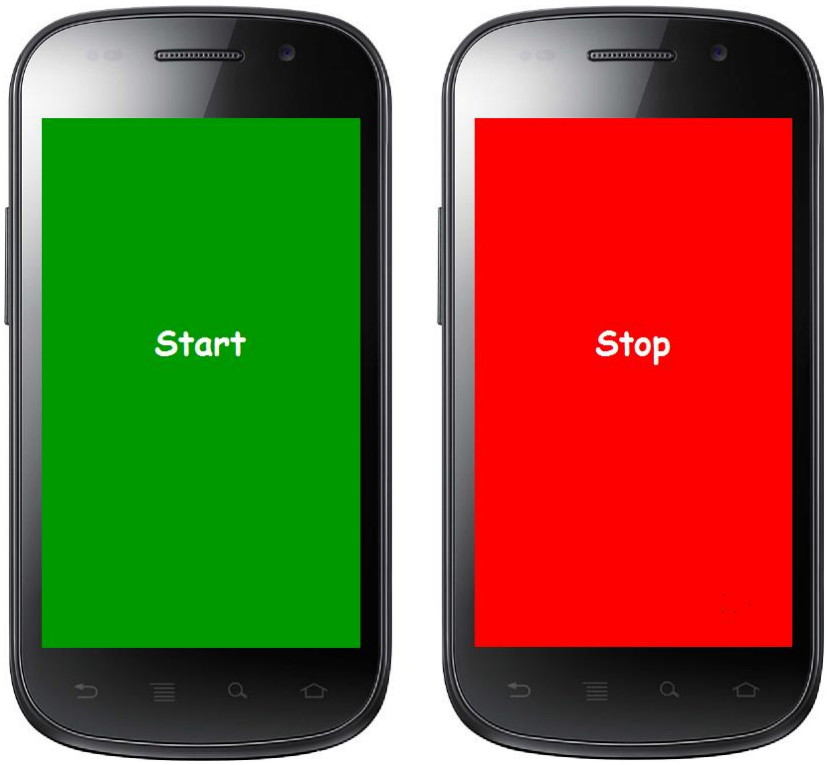
1. A felhasználó a kezdőképernyőről (vakoknak és gyengénlátóknak szánt felület) a beállítások felületre navigál.
2. A rendszer megjeleníti a beállítások képernyőt.
3. A felhasználó beállítja, hogy a telefon rezgetője és/vagy hangvezérléssel szeretne egyenesen haladni. Itt lehet az érzékenységen állítani, illetve a hangvezérlés típusán.

4. A rendszer végrehajtja a beállítást és elmenti az eszközön.

3.2. Felhasználói felület

A felhasználói felület az egyszerűségen és a könnyű értelmezhetőségen alapszik. A felületet a `Pencil` vázlatkészítő szoftver segítségével terveztem meg. A szoftver előnye, hogy ingyenes és egyszerűen használható. Nagyon könnyen bővíthető több felhasználható felületi elemmel.

Az alkalmazás két módban használható, ezért a látók számára készült felhasználói felület eltér a vakok és gyengénlátóknak szánt felülettől. Az alkalmazás indulásakor alapértelmezetten a vakok és gyengénlátóknak szánt felület jelenik meg **[8. ábra].** Természetesen a kezdőképernyő felülete akadálymentes, hogy minél egyszerűbben tudják kezelni vakok és gyengénlátók. A képernyő közepén alapértelmezetten egy *Start* feliratú gomb helyezkedik el, amelynek fel vagy le irányú végigsimításakor elindul a felhasználó irányítása egyenes vonal mentén. Ha a gombot újra végigsimítják ebben az irányban, akkor kikapcsol az irányítás és újból alapértelmezett állapotba kerül a felület. Erről az Activity-ről könnyen tovább tud navigálni egy látó felhasználó akár a látók számára elkészített Activity-re, vagy a beállításokba. Ehhez a lenyíló menü segítségével lehet navigálni.



8. ábra. Kezdőképernyő Activity-je

A kezdőképernyőről a beállítások Activity-re navigálva az alkalmazás működésével kapcsolatos beállításokat végezhetünk el, mint például legyen-e rezgetés, hangvezérléssel kapcsolatos beállítások stb. **[9. ábra].** A felület kezeléséhez látó felhasználó szükséges.

9. ábra. Beállítások Activity-je

Ha a felhasználó a látó felületet szeretné igénybe venni, akkor a kezdőablakból a menüben a Teszt megnyomására odanavigálhat. Ez az Activity valójában a tesztelésre szánt felület is egyben. A rendszer ezen keresztül tájékoztatja a felhasználót pl. a szenzor értékekről és egyéb információkról **[10. ábra]**. Ezen a felületen könnyen megfigyelhető, hogy helyes-e az alkalmazás működése. Itt is megtalálható egy Start gomb, amivel az egyenes vonalú haladás irányítása elindítható. Értelemszerűen a Stop gombbal tudja a felhasználó befejezni az irányítást. A látó felhasználó érdekében, de főleg a tesztelés miatt egy egyszerű iránytű is megtalálható ezen a felhasználói felületen. Ez csupán referenciaként szolgál a készülék mágneses szenzorjának az esetleges kalibrálására. A rendszer emellett kiírja a képernyőre a felhasználó által a Start gomb megnyomásakor elmentett irány értékét, az aktuális irány értékét, a tesztelés miatt az adott időközönkénti átlag értékékeket és, hogy melyik irányba tartson a felhasználó, hogy a helyes irányban haladjon. Természetesen erről a felhasználói felületről is el tud navigálni a felhasználó az alkalmazás beállításaiba a Beáll. gomb megnyomásakor.

10. ábra. Látóknak szánt felület

4. Implementáció

Ebben a részben implementálásra kerül az alkalmazás forráskódja. Az alkalmazás `Android Studio` fejlesztői eszközben került fejlesztésre. Ez egy integrált fejlesztői eszköz (integrated development environment - IDE) Android platformhoz. Elégséges és kellemes fejlesztői környezetet biztosít Android alkalmazások írásához, emellett rendelkezik grafikus felhasználói felület tervezővel is.

4.1. Szenzorok kezelése

Mindkét szenzornál (a gyorsulásmérőnél és a mágneses mező szenzornál) a mintavételezési frekvencia értékét be kellett állítani úgy, hogy a feladat szempontjából optimális legyen. Az Android által javasolt előredefiniált mintavételezési frekvenciákra konstansként lehet hivatkozni. Ezek az alábbiak:

- `SENSOR_DELAY_FASTEST` (0 ms)
- `SENSOR_DELAY_GAME` (20 ms)
- `SENSOR_DELAY_UI` (67 ms)
- `SENSOR_DELAY_NORMAL` (200 ms)

Természetesen a fejlesztő megadhat más értéket is. A feladathoz a 200 millisecundum mintavételezési frekvenciát választottam mindkét szenzor esetében. A felhasználó, ha neki úgy tetszik, beállíthatja ezt az értéket is a beállítások menüpont alatt. Ezeket az értékeket a szenzor listener-re való feliratkozáskor kell megadni:

```
sensorManager.registerListener(this, sensor,
SENSOR_DELAY_UI);
```

A gyorsulásmérő és a mágneses mező szenzor együtt használhatóak a készülék orientációjának meghatározásához. A két szenzorból érkező adatokból generálható a rotációs mátrix a `SensorManager.getRotationMatrix()` metódus hívásával. A generált rotációs mátrix átadható a `SensorManager.getOrientation()` metódusnak, ami kiszámítja a forgást (rotációt) az X,Y és Z tengelyek körül. Az Straight Line Walk alkalmazásban csak a Z tengely körüli értékek a fontosak. Ezt az értéket azimutnak vagy irányszögnek nevezik. A Z tengely a készülék felső részének az orientációját adja meg az északi irányhoz viszonyítva. Lényegében ezen alapszanak az iránytű alkalmazások.

4.1.1. Szűrők

A hibákkal számolni kell szenzorok adatainak feldolgozásakor. Ezeket a hibákat, eltéréseket le kell kezelni. Erre jó megoldás szűrők használata.

Az aluláteresztő szűrők (low-pass filters) kiszűrik a magas frekvenciájú jeleket vagy zajt. A felüláteresztő szűrők (high-pass filters) csak a magas frekvenciájú adatokat engedik át. A sáváteresztő szűrők (bandpass filters) nem engedik át se az alacsony, se a magas frekvenciájú adatokat. Csak a fejlesztő számára érdekes frekvenciatartományban tartja meg az adatokat. [Milette et al., 2012]

Az adatok elsimítására (smooth) való aluláteresztő szűrő egy általános megoldása magában foglalja a legújabb szenzorérték súlyozását a régivel szemben. Az *a* súlyozó paraméter használata:

*(Új érték) = (Előző érték) * (1– a) + x_i * a*

Java-ban ugyanez:

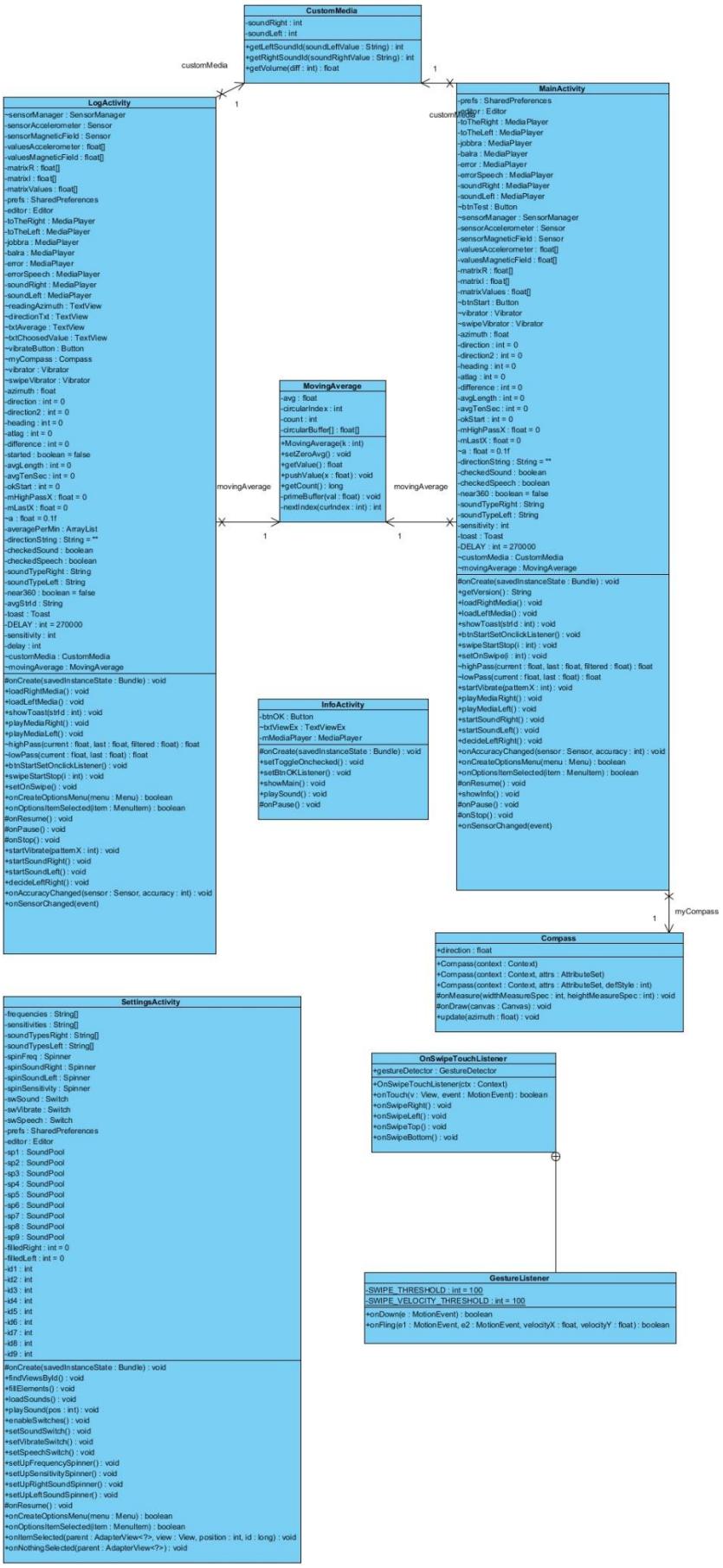
```
float a = 0.1f;

public void onSensorChanged(SensorEvent event
{
     x = event.values[0];
     mLowPassX = lowPass(x, mLowPassX);
}

// aluláteresztő szűrő
float lowPass(float aktualis, float elozo)
{
     return elozo * (1.0f - a) + aktualis * a;
}
```

4.2. Osztályok

Ebben a fejezetben az alkalmazás forráskódját mutatom be az osztályokon és a főbb metódusokon keresztül. A feladat osztálydiagramjában láthatóak az osztályok közötti kapcsolatok és az osztályok metódusai, illetve főbb attribútumaik. **[9. ábra]**



11. ábra. A feladat osztálydiagramja

4.2.1. MovingAverage

Az SMA (Simple Moving Average - Egyszerű Mozgó Átlag) módszer megtalálja a legutóbbi *k* adat értékeinek átlagát egy input folyamban. A *k* egész szám az átlagoló „ablak” méretét jelzi. A módszer akkor kezd működni, ha *k* darab érték beérkezett. [Milette et al., 2012]

A Straight Line Walk alkalmazásba az SMA módszert is beépítettem a jobb adatfeldolgozás érdekében. Később az SMA által visszaadott átlagértéket is figyelembe véve fogja a rendszer irányítani a felhasználót. Az SMA-t egy külön objektumként kezeltem. Az újonnan gyűjtött szenzor értéket a `pushValue()` metódussal adom át. A `getValue()` metódus meghívásával megkapom az átlagolt értéket.

4.2.2. Compass

Ez az osztály a View(nézet) osztály leszármazottja. A látó felhasználóknak szánt felületen (LogActivity) ez a nézet megjelenít egy iránytűt.

- `onDraw()`: ez a felüldefiniált metódus rajzol meg egy r sugarú kört és egy vonalat, amely a kör középpontjától a széléig ér. Az r sugár értéke függ a készülék képernyőjének méretétől. Ez azért kell, mert ha fix méretű lenne az r, akkor megtörténhet, hogy kicsi képernyőn kicsúszna a képernyőről ez az iránytű vagy nagy képernyőn túl kicsi lenne. Ha a képernyő nagyobb, mint 3,7 hüvelyk akkor r = képernyőSzélesség / 3.5, különben r = képernyőSzélesség / 2.5.
- `update()`: a paraméter alapján beállítja, hogy az `onDraw()` metódusban megrajzolt egyenes vonal a körön belül milyen irányban álljon. Tehát, hogy az iránytű a helyes irányba mutasson.

4.2.3. CustomMedia

Mivel az alkalmazás hangalapú visszacsatolásokat is ad a felhasználónak, ezért szükség van hangfájlok betöltésére is. A CustomMedia osztály egy példánya csupán a felhasználó által beállított hangfájlnak ad egy ID-t, ami alapján a MediaPlayer tudni fogja, hogy melyik hangfájlt indítsa el, ha kell.

- `getLeftSoundId()`: a metódus a paraméter alapján eldönti, hogy

melyik hangot választotta ki a beállításokban a felhasználó a balra való irányításhoz. Visszatérési értéke egy egész szám, ami a hangfájl ID-ja.

- `getRightSoundId()`: a `getLeftSoundId()` metódustól csak annyiban tér el, hogy ez a jobb oldali hangfájlnak ad ID-t.
- `getVolume()`: ez a metódus a jobbra és balra irányító hangfájlok hangerejét befolyásolja. Minél nagyobb az eltérés a kívánt iránytól, annál magasabb, illetve minél kisebb az eltérés annál alacsonyabb a hangerő.

4.2.4. OnSwipeTouchListener

Az OnSwipeTouchListener osztály egy példánya egy felületi elemen a fel/le és jobbra/balra simítás (swipe) gesztusokat kezeli le. Az osztály implementálja az OnTouchListener interfészt. A konstruktorban egy GestureDetector példány kerül létrehozásra. Ez az objektum észlel minden típusú képernyő érintést az eszközön. Ahhoz, hogy a GestureDetector végre is hajtson valamilyen műveletet egy ujjmozdulat hatására implementálni kell egy figyelőt (listener) az ujjmozdulatra definiált metódusokkal együtt. Ezért létrehoztam egy GestureListener belső osztályt, ami a SimpleOnGestureListener metódusait örökli. Ebben a belső osztályban felül kell definiálni az `onFling (MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)` metódust kötelezően azért, hogy a simítás gesztusokat lekezeljük. Ha az e1 és e2 objektumok X koordinátáinak a különbsége pozitív, akkor az `onSwipeRight()` metódus hívódik meg, különben az `onSwipeLeft()`. Ugyanígy, ha az e1 és e2 objektumok Y koordinátáinak a különbsége pozitív, akkor az `onSwipeBottom()`, különben az `onSwipeTop()` metódus hívódik meg. Az említett 4 meghívandó metódusok üres metódusok, amelyek majd csak akkor lesznek definiálva, amikor az osztályt példányosítjuk a MainActivity és LogActivity osztályokban.

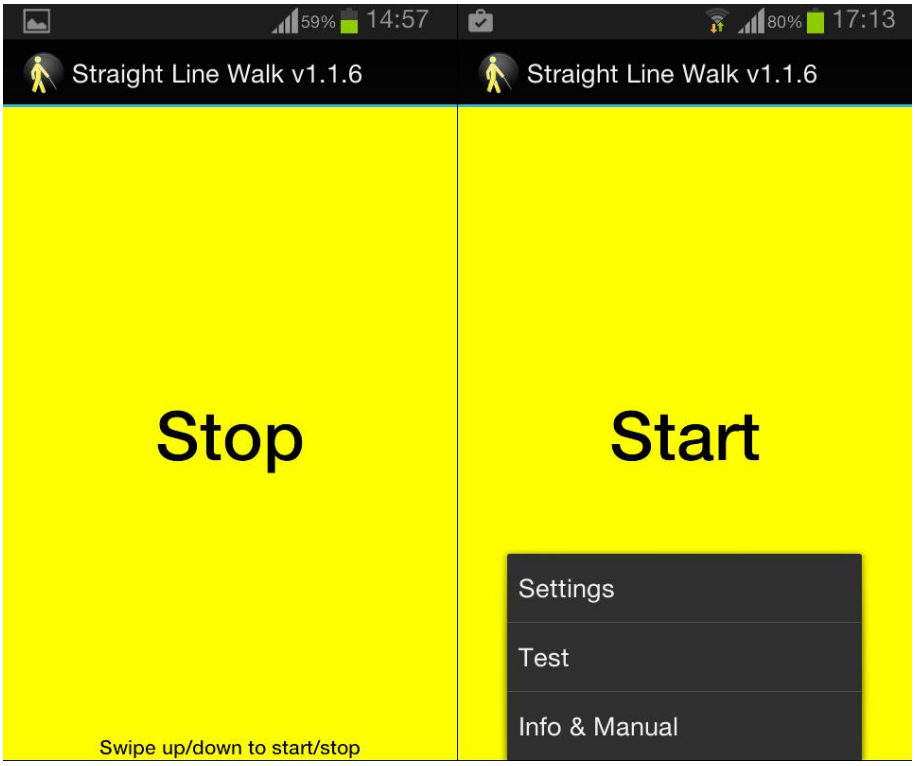
Ez a mintakód mutatja az OnSwipeTouchListener osztály használatát, egy grafikai elemre a képernyőn (pl. gomb, kép):

```
button.setOnTouchListener(new OnSwipeTouchListener() {
    public void onSwipeTop() {
        Toast.makeText(MyActivity.this,"fel",
Toast.LENGTH_SHORT).show();
    }
    public void onSwipeRight() {
```

```
        Toast.makeText(MyActivity.this,"jobbra",
Toast.LENGTH_SHORT).show();
    }
    public void onSwipeLeft() {
        Toast.makeText(MyActivity.this,"balra",
Toast.LENGTH_SHORT).show();
    }
    public void onSwipeBottom() {
        Toast.makeText(MyActivity.this,"le",
Toast.LENGTH_SHORT).show();
    }
});
```

4.2.5. MainActivity

Ez az Activity a főképernyőt és funkcióit valósítja meg. Ez a felület a vakoknak szánt képernyő **[12. ábra]**. A felület nagyon egyszerű, mivel csak egy az egész képernyőt kitöltő gomb található rajta. Az egyenes vonalú haladást úgy lehet indítani, hogy a felhasználó a képernyőn az ujját fel/le irányban végighúzza (megállítás ugyanilyen módon).



12. ábra. A főképernyő

- `onResume()`: itt regisztráljuk a listenert az adott szenzorokra a `registerListener()` beépített metódus segítségével.
- `btnStartSetOnClickListener()`: ez a metódus kezeli le az ujjal való simításokat (swipe) a gombon az OnSwipeTouchListener osztály segítségével. Mind a 4 irányt kezeli, meghívva a `swipeStartStop()` metódust a megfelelő paraméterrel. A fel vagy le irányban való simítás hatására a felhasználó egyenes vonalú irányítása indítható vagy fejezhető be. Jobbra vagy balra simításkor pedig az adott oldalra tud a felhasználó 90 fokkal fordulni.
- `swipeStartStop()`: indítás után a gomb felirata „Start”-ról „Stop”-ra vált és az egyenes vonalú haladás kívánt irányát elmenti egy változóba. Ha a jobbra vagy balra irányba húzta az ujját a felhasználó, akkor ennek a változónak az érékéhez hozzáad, vagy az értékéből kivon 90 fokot.
- `setOnSwipe()`: attól függően, hogy melyik irányba húzta az ujját a felhasználó a képernyőn, úgy hívódik meg ez a metódus a megfelelő paraméterrel. A metódus csak a gombon megjelenő feliratot kezeli.
- `loadRightMedia()`: a metódus betölti a jobb oldalra irányító csipogó hangot, amit a beállításokban kiválasztott a felhasználó. A MediaPlayer objektum a CustomMedia objektum segítségével kapja meg a hangfájl ID-ját.
- `loadLeftMedia()`: annyiban tér el az előző metódustól, hogy ez a baloldalra irányító csipogó hangot tölti be.
- `playMediaRight()`, `playMediaLeft()`: ha a beállításokban megengedett a csipogás az egyenes vonalú haladás irányítására, akkor a metódus elindítja a csipogást a megfelelő hangerővel (minél nagyobb az eltérés az eredeti iránytól annál hangosabb).
- `startSoundRight()`, `startSoundLeft()`: ez a két metódus dönti el, hogy a csipogás vagy a beszéd alapú hang induljon el. Ezt a beállítások alapján dönti el. Itt hívódnak meg a `playMediaLeft()`, illetve a `playMediaRight()` metódusok.
- `decideLeftRight()`: ez a metódus dönti el, hogy jobbra vagy balra irányítsa a felhasználót. A megfelelő feltételek mellett meghívódnak a

`startSoundRight()` és `startSoundLeft()` metódusok.

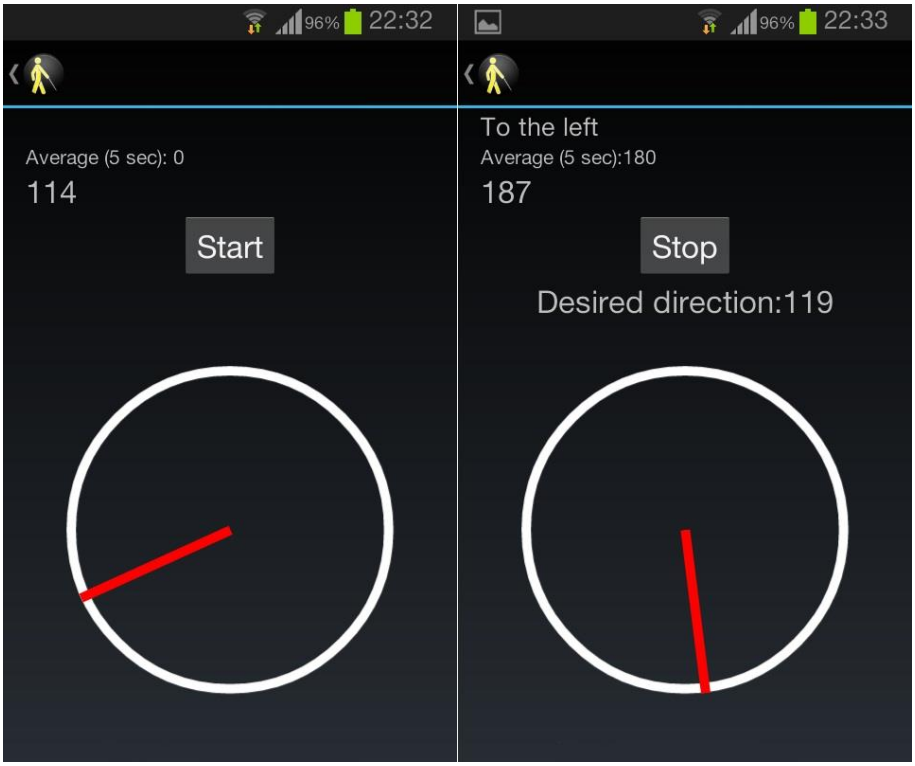
- `startVibrate()`: a paraméterül kapott érték (az aktuális irány eltérésének mértéke a kiválasztott iránytól) alapján egy rezgési mintát készít. Ha a vibráció, mint irányítási mód megengedett a beállításokban akkor a Vibrator objektum elkezdi a rezgetést a minta alapján.
- `highPass(), lowPass()`: felül- és aluláteresztős szűrők a szenzor értékek elsimításához.
- `onOptionsItemSelected()`: ezen felüldefiniált metódus segítségével a megnyíló menü egy elemének kiválasztása kerül lekezelésre. A menüből tovább lehet navigálni a beállításokba (SettingsActivity), látóknak szánt teszt felületre (LogActivity) és az alkalmazás leírásához (InfoActivity).
- `onSensorChanged()`: az osztály implementálja a SensorEventListener-t. Emiatt ez a metódus felüldefiniálásra kerül. A működés szempontjából ez az egyik legfontosabb metódus, amely akkor hívódik meg, ha valamelyik szenzor változást érzékel. Itt történik a szenzor adatok gyűjtése, és változásaik lekezelése. A gyorsulásérzékelő és a mágneses szenzor értékei alapján kiszámítódik az orientáció. Ehhez először kiszámítódik a rotációs mátrix a `SensorManager.getRotationMatrix()` metódus segítségével, amelyből aztán az orientáció is számítható a `SensorManager.getOrientation()` metódussal, ami egy 3 elemű tömböt ad vissza. A tömb első elemére van csak szükség, ami az azimut, vagyis az északhoz képesti irány, de radiánban. Az azimut értéket a további feldolgozás érdekében átalakítja fokra. Ebben a metódusban gyűjti be a MovingAverage objektum az átlagszámításhoz szükséges számú szenzoradatot. Bevetésre kerülnek az alul- és felülátereszto szűrők is. Itt kerül kiszámításra az aktuális irány (azimut) és a kiválasztott irány különbsége. Ha ez a különbség nagyobb egy bizonyos értéknél (beállításokban megadható az érzékenység), valamint az átlag 1 fokkal eltér a kiválasztott iránytól akkor a `startVibrate()` és a `decideLeftRight()` metódusok hívódnak meg.
- `onPause(), onStop()`: e két életciklus metódusban történik a

szenzor listener leiratkozása a szenzorokról az `unregisterListener()` metódussal. Ezt azért kell elvégezni itt, mert ha az Activity a Paused vagy Stopped fázisba kerül, ne használhassa feleslegesen a készülék akkumulátorát, mivel a szenzorok hosszútávú használata könnyen lemerítheti azt.

4.2.6. LogActivity

Ez az Activity a látó felhasználóknak szolgáló felületet biztosítja. Itt érdemes tesztelni az alkalmazás helyes működését, és értékeinek pontosságát. A LogActivity működésében csak annyiban különbözik a MainActivity-től, hogy a képernyőn megtalálható egy iránytű is, amelyet a Compass objektum reprezentál, és neki adódik át az aktuális irány értéke. Ezenkívül az alkalmazás szövegesen közli a kiválasztott irány értékét fokban, az aktuális irányt, az átlagot és azt, hogy melyik oldalra tartson a felhasználó, ha esetleg kitérne a helyes irányból. Az irányítás a Start gomb megnyomása után történik. **[13. ábra.]**

Az implementált metódusok csak annyiban térnek el a MainActivity metódusaihoz képest, hogy a képernyőn közlik az adatokat.

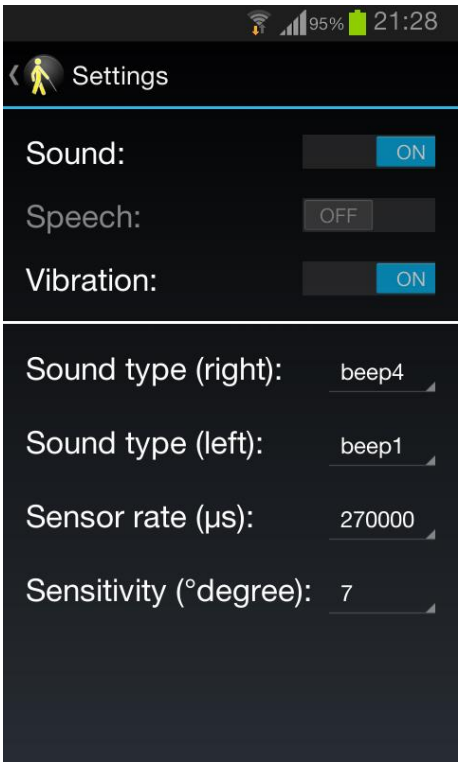


13. ábra. Látóknak szánt teszt felület (LogActivity)

4.2.7. SettingsActivity

A SettingsActivity az alkalmazás testreszabására, beállítására szolgáló felületet biztosítja **[14. ábra]**. Ezen a felületen lehet változtatni, hogy milyen jellegű visszacsatolást adjon a felhasználónak, ha eltér a helyes iránytól. Váltani lehet rezgés és/vagy hang alapú visszacsatolás között. Hangalapú visszacsatolás közül a beszédalapút és a csipogást lehet választani. Ezenkívül beállítható a csipogás típusa is külön a jobb és baloldalra is. Továbbá itt lehet beállítani a szenzorok érzékenységét és a mintavételezési frekvenciát. Az itt beállított értékek az egész alkalmazásra ki lesznek terjesztve.

A beállított értékek tárolására a SharedPreferences osztályt használtam. Ennek segítségével kulcs-érték párokat lehet könnyen tárolni és lekérdezni. Lényegében egy SharedPreferences objektum kulcs-érték párokat tartalmazó fájlra mutat és metódusokat szolgáltat az írásukra és olvasásuskra [12].



14. ábra. A beállítások képernyője (SettingsActivity)

- `fillElements()`: a metódus betölti a SharedPreferences objektum által elmentett fájlból az értékeket és ezekkel feltölti a megfelelő grafikai elemeket.
- `loadSounds()`: a csipogó hangok betöltése ebben a metódusban

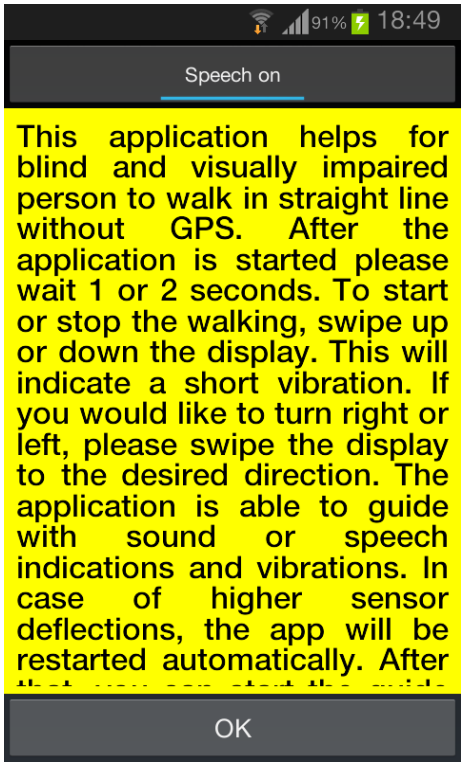
történik. Erre azért van szükség, mert ha a felhasználó a jobb vagy baloldal csipogását szeretné beállítani, akkor a lenyíló ablakban rákattintva a kívánt csipogásra elindul a hangfájl. A hangfájl lejátszása most a SoundPool osztály segítségével történik. Itt azért választottam a SoundPool osztályt, mert nincs szükség a csipogás menet közbeni leállítására, amit csak a MediaPlayer osztállyal lehetne megtenni. A SoundPool az alkalmazás teljesítménye szempontjából is jó, ugyanis használatával a készülék memóriájában tárolódik el az adott hangfájl, ahonnan gyorsabban kiolvasható. Persze ez csak kisméretű, rövid hangoknál előnyös.

- `playSound()`: itt történik annak az eldöntése, hogy melyik csipogás játszódjon le.
- `enableSwitches()`: ez a metódus kezeli le azt, hogy a csipogás és beszéd alapú visszacsatolást egyszerre ne lehessen beállítani.
- `setUpFrequencySpinner()`,`setUpSensitivitySpinner()`,`setUpRightSoundSpinner()`,`setUpLeftSoundSpinner()`: e metódusok szolgálnak az egyes lenyíló listák (Spinner) értékekkel való feltöltése.
- `onItemSelected()`: lenyíló listák használatakor a metódus felüldefiniálása kötelező. A metódus lekezeli azt az eseményt, ha a lista valamelyik elemét kiválasztják és a SharedPreferences objektum segítségével elmentődik a kiválasztott elem értéke.

4.2.8. InfoActivity

Ez az Activity azt a képernyőnézetet reprezentálja, ami az alkalmazás telepítése utáni első indításkor megjelenik a felhasználónak, de a menüből később is megnyitható. A **15. ábrán** látható módon ez az Activity arra szolgál, hogy a felhasználónak elmagyarázza az alkalmazás működését szövegesen a képernyőre írva és hangos beszéddel is. A hangot a képernyő felső részén található kapcsolóval lehet kikapcsolni. A képernyő alján az OK gombra kattintva a MainActivity főképernyőre jutunk. A leírás betűméretét a konvencióktól eltérően nagyra méreteztem a gyengénlátókra gondolva. A szöveges leírás görgethető nézetben van (ScrollView).

- `setToggleOnChecked()`: itt a ToggleButton kapcsolóra készít egy listenert és lekezeli a kapcsoló On/Off eseményeit. Ha On üzemmódban van, akkor a `playSound()` metódus hívódik meg, különben a MediaPlayer objektum által elindított hangfájl leáll.
- `playSound()`: a metódus a beszéd alapú tájékozató hangfájlját indítja el, ha a képernyő felső részén lévő kapcsoló bekapcsolt állapotban van. A hangfájl a MediaPlayer osztály segítségével indul el.
- `showMain()`: a `playSound()` metódusban a hangfájl lejátszása után ez a metódus hívódik meg. Létrehoz egy Intent-et, amely aztán a MainActivity kezdőképernyőjét nyitja meg.



15. ábra. InfoActivity képernyőképe

4.2.9. TextViewEx és TextJustify osztályok

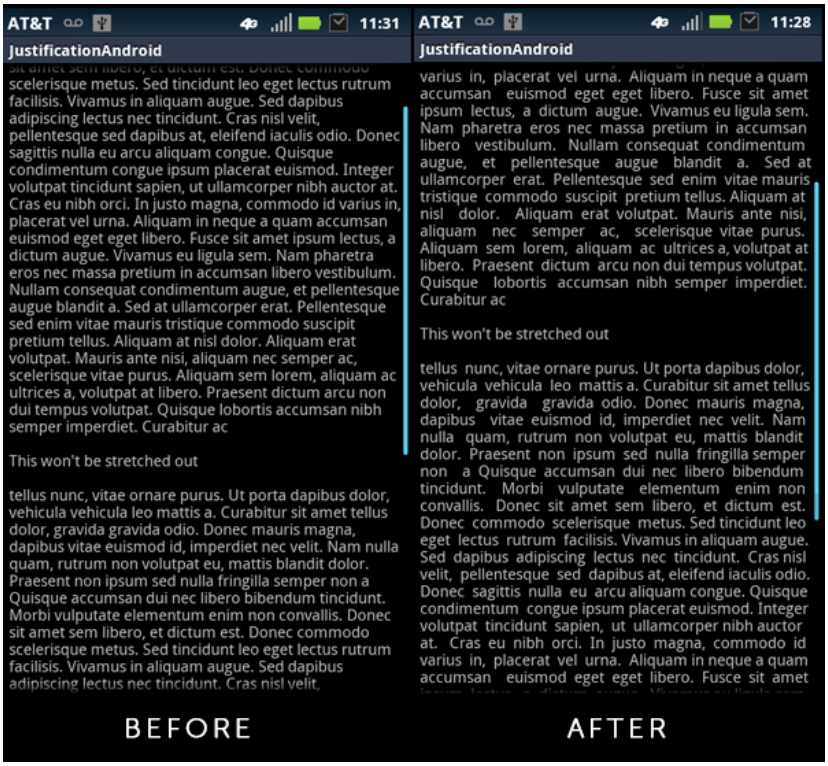
Ez a két osztály előre elkészített osztály a GitHub oldalról.[13] Azért került bele a forráskódba, mivel az Androidban nincs olyan beépített opció, amivel szövegeket sorkizárttá lehetne tenni. Az InfoActivity-ben használtam fel.

Ugyanolyan attribútumai vannak, mint egy egyszerű TextView-nak. Az elrendezés (layout) XML kódjában a TextView helyett ilyen módon lehet megvalósítani:

```
<com.textjustify.TextViewEx
        android:id="@+id/textview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

A Java forráskódban pedig:

```
TextViewEx txtViewEx = (TextViewEx)
findViewById(R.id.textViewEx);
txtViewEx.setText("Insert your content here", true); //
true: sorkizárást engedélyezi
```



16. ábra. Egy szöveg nézete TextJustify nélkül és utána

4.3. Erőforrások

Az alkalmazásban használt hangfájlokat egy erőforrás mappában tároltam. A csipogó hangokat a http://www.soundjay.com/beep-sounds-1.html weboldalról szereztem be. A beszéd alapú hangokat a Google Fordító segítségével oldottam meg. Egy URL sablonba ágyazott szöveget kell csak beírni a böngészőbe, és beállítani a nyelvet is (magyar-hu, angol-en) az alábbi módon:

http://translate.google.com/translate_tts?tl=hu&q="helló%20világ!".
Ezek után a hangfájl letölthető. Az útmutató hangfájl hossza miatt szükség volt mondatokra szedni a szöveget. Minden mondat egy fájl lett, amiket aztán az `MP3Wrap` programmal fűztem össze.

4.4. Az alkalmazás disztribúciója

A Straight Line Walk alkalmazás hivatalosan 2014. október 6-án került feltöltésre a legnagyobb Android alkalmazásokra specializálódott elektronikus áruházba, a Google Play Áruházba.[14] A Google Play Áruház, eredetileg Android Market, egy digitális tartalmakat árusító, webes áruház. Lehetőség van még vásárolni és letölteni az alkalmazásokon túl dalokat, filmeket, esetleg könyveket is. A Play Áruházon kívül léteznek kisebb alternatív áruházak Android alkalmazásokra specializálódva, azonban ennek vannak a legnagyobb előnyei. Kliens alkalmazásként a készülékek többségén az alkalmazások elsődleges forrása. Az alkalmazások beszerzése és aktualizációja is nagyon egyszerű vele. A Google Play több, mint egymillió alkalmazást tartalmaz, aminek többsége ingyenesen elérhető. [15]

4.4.1. Az alkalmazás feltöltése a Google Play Áruházba

Egy alkalmazás feltöltéséhez szükség van egy fejlesztői fiók elkészítésére a *play.google.com/apps/publish* weboldalon. Azért, hogy a fejlesztők ne oszthassanak meg akármit, a Google bevezetette a 25$-os egyszeri regisztrációs díjat. A közzététel előtt szükséges az alkalmazást becsomagolni és feltölteni a Google Play-be.

Az Android platform megköveteli, hogy minden telepített alkalmazás legyen digitálisan aláírva egy bizonylattal, aminek a titkos kulcsa a fejlesztő tulajdonában van. Az alkalmazás feltöltése előtt ez a kulcs ellenőrizve van a Google Play-en. Ha az alkalmazás nincs aláíratva az SDK-val (általában a debug.keystore bizonylattal) nem lehet feltölteni az alkalmazást. A bizonylat egy eszköz az alkalmazás fejlesztőjének azonosítására, ezért ajánlott a saját alkalmazásainkat ugyanazzal a bizonylattal ellátni. Az Android Studio és az Eclipse fejlesztőkörnyezetek biztosítanak eszközt a bizonylat generálására és aláírására.

Az alkalmazás feltöltése előtt a projekt Manifest fájljában le kell tiltanunk a hibakaresést(debug). Az eljárás a következő Android Studio-ban: megnyitjuk a projektet, amelyet szeretnénk becsomagolni és aláíratni, kiválasztjuk a felső menüben a *Build* → *Generate Signed APK*... opciót. Beírjuk a mesterjelszót, amelyet a fejlesztő megadott

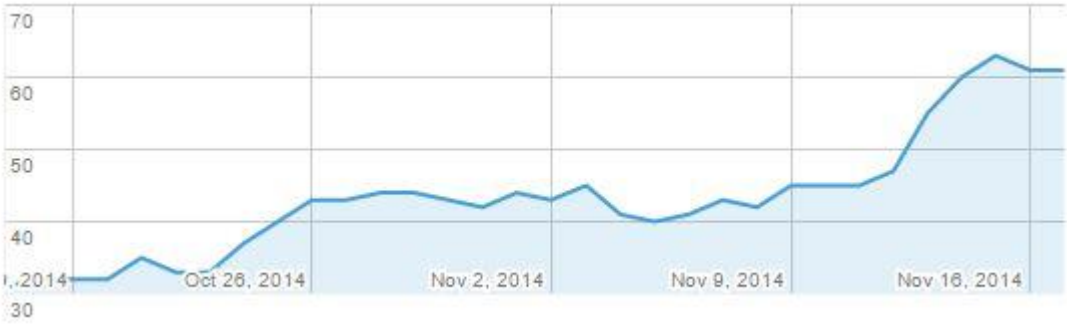
magának. Ha még nincs titkos kulcsunk, akkor itt lehet készíteni a *Create new…* gombra kattintva. Készíteni kell egy úgynevezett alias-t, ami után meg kell adni egy jelszót, az érvényesség idejét (alapértelmezetten 25 év) és személyes információkat. Ha már létezik titkos kulcsunk, akkor a kulcsot tároló fájl elérési útját kell csak megadni az alias és a jelszó mellett. Az Ok gombra kattintva kiválasztható, hogy hova mentse a rendszer az elkészített .apk fájlt. Ezt a fájlt még optimalizálni kell a Zipalign.exe segítségével, amely az Android SDK-val együtt feltelepült a fejlesztő számítógépére. Ezt a programot a parancssorból tudjuk indítani a `zipalign [-f] [-v] 4 infile.apk outfile.apk` parancssal. Ezekután az .apk fájl feltölthető a Google Play-be.

Az alkalmazás feltöltésekor meg kell adni az alkalmazás nevét, elsődleges nyelvét. Utána kitölthetünk mezőket, amelyek információkat fognak közölni az alkalmazásról (pl. leírás, képek, videók). Ezeket az információkat lokalizálva is meg lehet adni, egy másik nyelv hozzáadásával. A Straight Line Walk alkalmazás alapértelmezetten angol, de magyar lokalizáció esetén magyar nyelvű. A liszenszfeltételek elfogadása után közzétehető az alkalmazás.

4.4.2. Statisztikák

Az alkalmazás feltöltése után természetes elvárás az, hogy követni lehessen, hogyan és kik töltik le azt. A Google Play Áruház fejlesztői részébe bejelentkezve a felhasználó fiókban megtekinthetőek ezek a statisztikák. Megtekinthető a letöltések száma, a jelenleg feltelepített alkalmazások száma, napi letöltések stb. Ezeket az adatokat lehet szűrni több kritérium alapján is, mint például Android verzió, készülék típusa, ország, nyelv stb. Ezen a felületen lehet követni az esetleges hibákat, amik miatt leállt az alkalmazás valamelyik felhasználónál. Ezeken kívül követhető a felhasználók értékelése és hozzászólásaik is.

A Straight Line Walk alkalmazás jelenleg (2014. november 18.) 61 készülékre van feltelepítve. A felhasználók elég sokszínűek. Magyarországon kívül a legtöbb letöltő az Indonéziából (16) és az USA-ból (13) van, de Chiléből, Irakból és még Namíbiából is vannak felhasználók. Az Android verziókat tekintve a legtöbb felhasználó (23) a 4.4-es verziót használja.



17. ábra. Aktuális telepítések száma (2014 október 12.-november 18.)

18. ábra. Az aktuális telepítések Android verziókra levetítve (2014. november 10.)

5. Tesztelés

Az alkalmazást 10 emberen teszteltük „terepen”, amely egy ugyanolyan hosszú út (20m) megtétele volt. A tesztelés során a jobbra és balra fordulásokat is belevettük. Tehát az út során egyszer kellett balra is és jobbra is fordulnia a felhasználónak 90 fokkal. A tesztelés bekötött szemű látó felhasználókkal történt. **[19. ábra]**

19. ábra. Tesztelés "terepen"

A tesztelés után a tesztelők mindannyian kitöltöttek egy kérdőívet, amelyekben az alkalmazás használhatóságáról kaptak állításokat, amikkel vagy egyet értettek vagy nem. Az állítások az alábbiak voltak:

1. A hangjelzés hangereje kellemes volt.
2. A hangerősség megfelelő volt a rendszer visszacsatolásainak megértéséhez.
3. Idegesítőnek találtam azt, hogy minél jobban kitérek a helyes iránytól egyre hangosabb a hangjelzés.
4. Úgy gondolom, hogy a hangalapú visszacsatolás segített az egyenes haladásban.
5. Magabiztosságot éreztem a rendszerrel szemben.
6. Fárasztó lenne hosszabb ideig használni a rendszert.
7. A hangalapú visszacsatolás zavart a mindennapos tájékozódásban.

8. Érdekelne a rendszer napi használat miatt.
9. Úgy érzem, hogy a rendszer segítene gyakorolni, hogy egyenesen tudjak haladni.

Az eredményeket a következő táblázat tartalmazza:

2. táblázat. A tesztelés utáni felmérés eredménye

	1. állítás	2. állítás	3. állítás	4. állítás	5. állítás	6. állítás	7. állítás	8. állítás	9. állítás
1. tesztelő	4	4	2	4	3	1	3	3	3
2. tesztelő	3	4	2	4	4	3	2	4	3
3. tesztelő	2	4	1	3	2	2	2	3	3
4. tesztelő	4	4	2	4	3	2	2	4	3
5. tesztelő	3	4	2	4	3	2	2	4	3
6. tesztelő	3	4	1	3	2	3	3	3	3
7. tesztelő	2	4	1	3	4	2	2	4	3
8. tesztelő	3	4	2	4	4	1	2	4	3
9. tesztelő	4	4	1	4	3	2	2	4	3
10. tesztelő	3	3	2	4	2	2	3	3	3
Átlag	3,1	3,9	1,6	3,7	3	2	2,3	3,6	3

1 = Egyáltalán nem ért egyet 2 = Nem ért egyet 3 = Egyetért 4 = Teljesen egyetért

A terepen való tesztelésnél a 20 méteres út megtétele után (menet közben jobbra és balra is fordulva) átlagosan 20 cm-es eltérés volt az eredeti iránytól, ha az érzékenységet 7 fokos-ra állítottuk.

5.1. Tesztesetek

Ebben a fejezetben az alkalmazás egyéb funkcióinak tesztesetei kerülnek felsorolásra. A felhasználói felület kezelésével kapcsolatos funkciók kerülnek tesztelésre.

3. táblázat. Tesztesetek

Leírás képernyő			
Funkció	**Lépések**	**Elvárt eredmény**	**Eredmény**
Beszéd kikapcsolása	1. A felhasználó megnyomja a felső sávban a kapcsoló gombot.	A beszéd alapú magyarázat kikapcsol.	Igazolva
Főképernyőre navigálás	1. Az OK gombot megnyomja a felhasználó. 2. Az alkalmazás elindítja a leírás képernyőt.	Megjelenik a főképernyő (vakoknak szánt felület).	Igazolva
Főképernyő			
Egyenes vonalú haladás indítása	1. A felhasználó le vagy fel irányban végigsimítja a képernyőt az ujjával a kiválasztott irányba tartva a készüléket. 2. A felirat „Stop”-ra vált. 3. Az alkalmazás egy apró vibrációval jelez. 4. Az alkalmazás elindítja a felhasználó irányítását.	Elindul az egyenes vonalú haladás irányítása.	Igazolva
Egyenes vonalú haladás leállítása	1. A felhasználó le vagy fel irányban végigsimítja a képernyőt az ujjával. 2. A felirat „Start”-ra vált. 3. Az alkalmazás egy apró vibrációval jelez. 4. Az alkalmazás leállítja a felhasználó irányítását.	Kikapcsol az egyenes vonalú haladás irányítása.	Igazolva
Jobbra fordulás	1. A felhasználó jobbra húzza az ujját a képernyőn. 2. Az alkalmazás 90 fokkal a kiválasztott iránytól jobbra irányít.	Az alkalmazás az előzőleg kiválasztott iránytól 90 fokkal jobbra irányítja a felhasználót.	Igazolva

Főképernyő			
Balra fordulás	1. A felhasználó balra húzza az ujját a képernyőn. 2. Az alkalmazás 90 fokkal a kiválasztott iránytól balra irányít.	Az alkalmazás az előzőleg kiválasztott iránytól 90 fokkal jobbra irányítja a felhasználót.	Igazolva
Menü	1. A felhasználó megnyomja a menü gombot.	Felnyílik a menü	Igazolva
Beállítások	1. A felhasználó a menüből kiválasztja a „Beállítások”-at. 2. Az alkalmazás elindítja a beállítások képernyőt.	Elindul a beállítások képernyő.	Igazolva
Teszt	1. A felhasználó a menüből kiválasztja a „Teszt”-et. 2. Az alkalmazás elindítja a látóknak szánt teszt képernyőt.	Elindul a látóknak szánt teszt képernyő.	Igazolva
Leírás	1. A felhasználó a menüből kiválasztja a „Leírás”-t. 2. Az alkalmazás elindítja a leírás képernyőt.	Elindul a leírás képernyő.	Igazolva
Teszt képernyő			
Visszanavigálás a főképernyőre	1. A felhasználó a felső sávban levő ikont megnyomja. 2. Az alkalmazás újból elindítja a főképernyőt.	Elindul a főképernyő.	Igazolva
Egyenes vonalú haladás indítása	1. A felhasználó megnyomja a „Start” gombot 2. A felirat „Stop”-ra vált. 3. Az alkalmazás elindítja a felhasználó irányítását.	Elindul a felhasználó egyenes vonalú haladásának irányítása a kiválasztott iránynak megfelelően.	Igazolva

Teszt képernyő			
Egyenes vonalú haladás leállítása	1. A megnyomja a „Stop” feliratú gombot 2. A felirat „Start”-ra vált. 3. Az alkalmazás leállítja a felhasználó irányítását.	Kikapcsol az egyenes vonalú haladás irányítása.	Igazolva
Menü	1. A felhasználó megnyomja a menü gombot.	Felnyílik a menü	Igazolva
Beállítások	1. A felhasználó a menüből kiválasztja a „Beállítások”-at. 2. Az alkalmazás elindítja a beállítások képernyőt.	Elindul a beállítások képernyő.	Igazolva
Leírás	1. A felhasználó a menüből kiválasztja a „Leírás”-t. 2. Az alkalmazás elindítja a leírás képernyőt.	Elindul a leírás képernyő.	Igazolva
Beállítások képernyő			
Visszanavigálás	1. A felhasználó a felső sávban levő ikont megnyomja. 2. Az alkalmazás újból elindítja az előző képernyőt.	Elindul az előzőleg megnyitott képernyő.	Igazolva
Vibráció	1. A felhasználó az ide tartozó kapcsolót megnyomja. 2. Az alkalmazás menti a beállítást.	Az alkalmazás menti a beállítást és bekapcsol a rezgetés az egyenes vonalú haladás irányításakor.	Igazolva
Csipogás	1. A felhasználó a kapcsolót megnyomja. 2. A beszéd kapcsoló inaktívvá válik. 3. Az alkalmazás menti a beállítást.	Az alkalmazás menti a beállítást és bekapcsol a csipogás az egyenes vonalú haladás irányításakor.	Igazolva

Beállítások képernyő			
Beszéd	1. A felhasználó a kapcsolót megnyomja. 2. A csipogás kapcsoló inaktívvá válik. 3. Az alkalmazás menti a beállítást	Az alkalmazás menti a beállítást és bekapcsol a beszéd az egyenes vonalú haladás irányításakor.	Igazolva
Csipogás típusa jobbra	1. A felhasználó a lenyíló listából.kiválasztja a hangtípust. 2. A kiválasztott csipogás lejátszódik. 3. Az alkalmazás menti a beállítást.	Az alkalmazás menti a beállítást.	Igazolva
Csipogás típusa balra	1. A felhasználó a lenyíló listából.kiválasztja a hangtípust. 2. A kiválasztott csipogás lejátszódik. 3. Az alkalmazás menti a beállítást.	Az alkalmazás menti a beállítást.	Igazolva
Szenzor mintavételezési frekvenciája	1. A felhasználó a lenyíló listából.kiválasztja az értéket. 2. Az alkalmazás menti a beállítást.	Az alkalmazás menti a beállítást.	Igazolva
Szenzor érzékenysége	1. A felhasználó a lenyíló listából.kiválasztja az értéket. 2. Az alkalmazás menti a beállítást.	Az alkalmazás menti a beállítást.	Igazolva

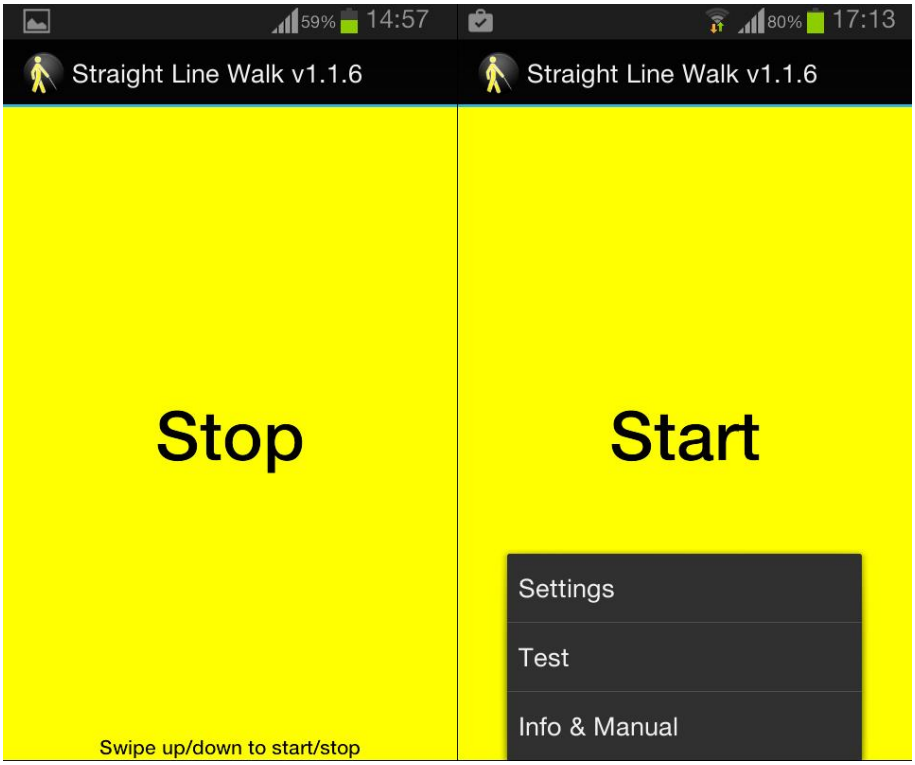
6. Felhasználói dokumentáció

A Straight Line Walk egy vakoknak és gyengénlátóknak szánt alkalmazás. A felhasználó egyenes vonalú haladását segíti. A rendszer rezgésekkel és/vagy hangokkal tudja irányítani a felhasználóját. Az alkalmazás használatához szükséges, hogy a telefon tartalmazzon mágneses szenzort és gyorsulásérzékelőt.

Az alkalmazás felületét tekintve nem túl látványos, mivel vakoknak és gyengénlátóknak készült akadálymentes a felület. A látó felhasználóknak is készült egy felület tesztelés érdekében. Az alkalmazásban a menü segítségével lehet navigálni. A kezdőképernyőre való visszanavigálás a felső sávban az alkalmazás ikonjára kattintva lehet.

6.1. Főképernyő

A főképernyő a **20. ábrán** látható módon jelenik meg. Ez a vakoknak és gyengénlátóknak szánt felület. A felület akadálymentesített, ezért a sárga/fekete színkombináció. Ezen a képernyőn maradva lehet indítani az egyenes vonalú haladás irányítását.



20. ábra. Vakoknak szánt főképernyő felülete

Az egyenes haladás irányításának indításához a kívánt irányba kell beállnia a felhasználónak. Ezután a képernyő felfele vagy lefele irányban történő végigsimításával lehetséges az indulás. Ha ezt a mozdulatot helyesen hajtja végre a felhasználó, akkor ezt egy kis vibráció jelzi és a „Start” helyett a „Stop” felirat jelenik meg. Ezután a felhasználó indulhat egyenesen előre. A készüléket a **21. ábrán** látható módon, egyenesen előre a képernyőt felfele fordítva kell tartani a kézben. Ahhoz, hogy egyenes irányban haladjon a felhasználó, követnie kell a készülék visszacsatolásait és reagálni rájuk. Ezek a visszacsatolások a beállítások alapján lehetnek rezgések, beszéd, illetve csipogás. Minél jobban kitér a kívánt iránytól annál intenzívebb a rezgések sűrűsége, illetve annál hangosabb a hang alapú visszacsatolás. Ha 90 fokkal oldalra szeretne fordulni, akkor az adott irányba kell, hogy húzza az ujját, amit ismét egy kis rezgés jelez.

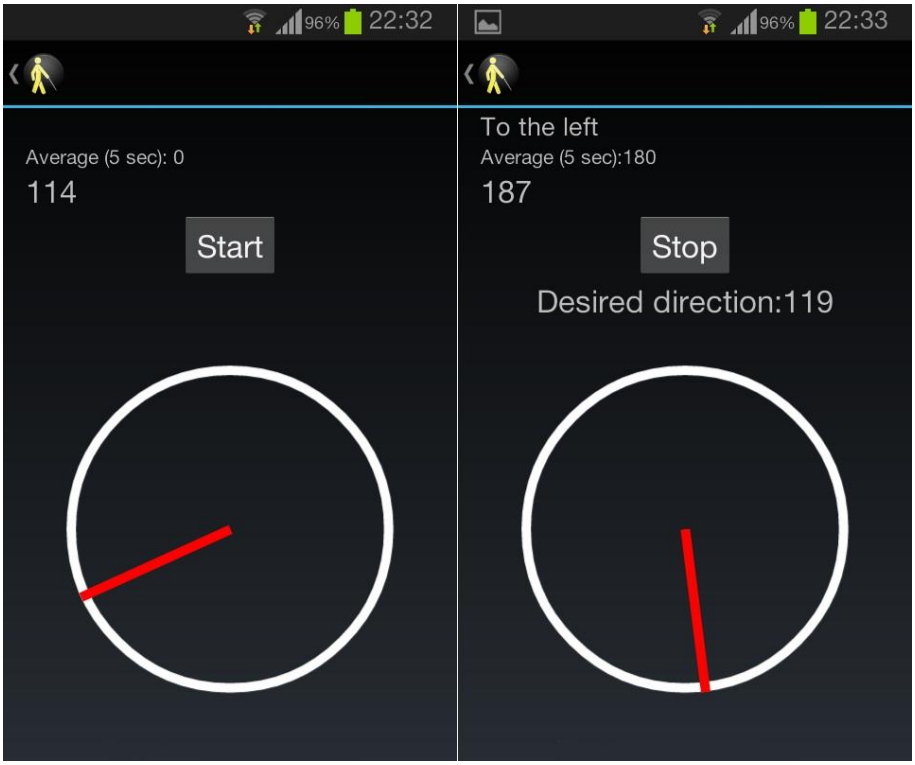
21. ábra. A készülék helyes módon tartása a kézben

A menü gombot megnyomva (Android 4.4-től felülről lenyíló menü) a készüléken megjelennek lehetőségek további felületekre való navigációra (pl. beállítások).

6.2. Teszt képernyő látóknak

A menüből kiválasztva a Teszt felületre lehet jutni. Az alkalmazás tesztelésére, illetve helyes működésének ellenőrzésére látó felhasználóknak ezen a képernyőn van lehetőség. A felület képernyőképe a **22. ábrán** látható. Ezen a felületen egy iránytű is található, hogy látható legyen az adott irány, amelynek számértékét és átlagát fokban szintén kiírja a rendszer. Működésben ez a felület semmiben nem tér el a főképernyőtől, csak itt a

számértékek is megfigyelhetőek. A Start gomb megnyomásakor indítható az egyenes vonalú haladás, amely alatt megjelenik a kiválasztott irány. A bal felső sarokban pedig, hogy melyik irányba tartson menet közben a felhasználó. A főképernyőre navigálni a felső sáv bal oldalán levő ikonra kattintva lehet. A menü segítségével pedig a beállításokba és az alkalmazás működésének leírásához lehet navigálni.



22. ábra. Látóknak szánt teszt felület

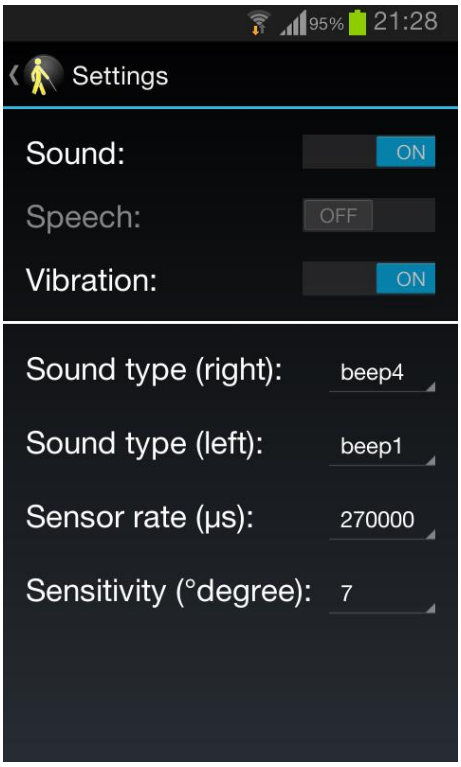
6.3. Beállítások

A menüből a beállításokba navigálhat a felhasználó, ahol egy látó felhasználó segítségével beállításokat végezhet a rendszerhez. A **23. ábrán** látható, hogy milyen lehetőségek vannak a felületen a beállításokra. Az ezen a felületen elvégzett beállítások érvényesek lesznek az egész rendszerre.

Az első szekcióban kapcsoló gombok találhatóak, melyek segítségével azt lehet beállítani, hogy a rendszer milyen módon adjon visszacsatolást a felhasználónak az egyenes vonalú haladás irányításához. Az első két kapcsoló a hangalapú visszacsatolás választására szolgál. Itt lehet változtatni, hogy beszédalapú vagy csipogó hanggal segítse a felhasználót az egyenes haladásban a rendszer. Egyszerre csak az egyik mód lehet aktív. A harmadik kapcsoló a vibráció ki- és bekapcsolására szolgál. Alapértelmezetten a beszédalapú jelzés és

a vibráció van bekapcsolva.

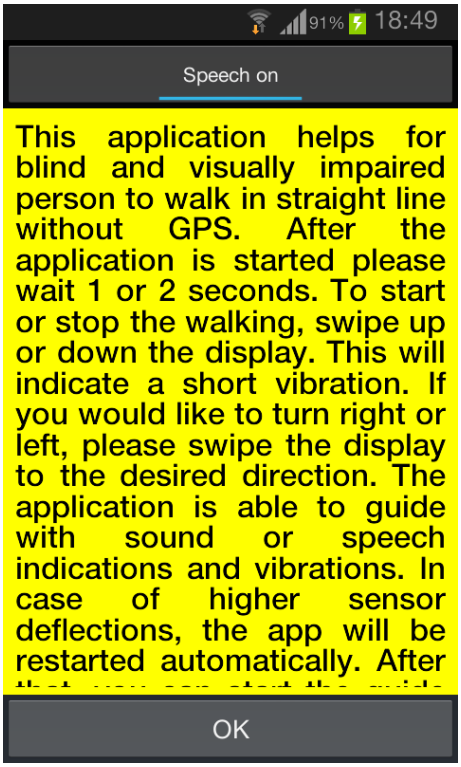
A második szekcióban lenyíló listák találhatóak. Az első kettőben a jobb- és baloldalra irányító csipogás típusát lehet kiválasztani. A harmadikban a szenzorok mintavételezési frekvenciáját lehet kiválasztani mikroszekundumokban (alapértelmezetten 270000 µs). Ezalatt található az érzékenységet befolyásoló lenyíló lista. Itt állítható be, hogy hány fokos eltérésnél jelezzen a rendszer, ha kitérnek a kiválasztott irányból (alapértelmezetten 7 fok). Utóbbi kettőre a sokféle típusú készülék és szenzor miatt van szükség. A főképernyőre navigálni a felső sáv bal sarkában levő ikonra kattintva lehetséges.



23. ábra. A beállítások képernyője

6.4. Leírás képernyő

Az alkalmazás telepítése után az első indításkor ez a felület jelenik meg a főképernyő előtt. A **24. ábrán** látható, hogy ez a felület csupán egy leírást tartalmaz az alkalmazás használatáról. A főképernyőről a menü segítségével ugyanúgy lehet ide is navigálni. Alapértelmezetten a felhasználót beszéd is alaposan tájékozatja az alkalmazás használatáról, de a hang a képernyő felső sávjában a kapcsoló gombbal kikapcsolható. Ha a hangot végig lejátszotta a rendszer a felület automatikusan a kezdőképernyőre vált. Az OK gombbal is a főképernyőre lehet navigálni.



24. ábra. A leírás képernyője

7. Konklúzió

7.1. A projekt értékelése

A projekt során egy Google Android alkalmazás tervezésére és fejlesztésére került sor, amely képes a vakok és/vagy gyengénlátók egyenes vonalú haladását segíteni. Vak embereknek a sok gyakorlás ellenére is nehézséget okoz az egyenes haladás. A Straight Line Walk alkalmazás ennek a problémának a csökkentésére készült, az okostelefon valósidejű hang- és rezgésalapú visszacsatolásainak nyújtásával. A tesztelések során kiderült, hogy jó megoldásnak bizonyult az az elgondolás, miszerint az egyenestől való kitérés mértékének függvényében erősödik vagy gyengül a hang alapú visszacsatolás. Az alkalmazásban való műveletekhez kapcsolódóan arra jutottam, hogy ne gombok legyenek a képernyőn, hanem a felhasználó az ujjával simítsa végig a képernyőt bizonyos műveletek végrehajtásához, mert egy vak felhasználó nehezen találja meg a gombokat a képernyőn.

A rendszer szempontjából kritikus a mágneses szenzor, amely bizonyos tárgyak közelében (pl. vasoszlop, transzformátor, mágnes) megzavarodhat és helytelen értékeket nyújthat a rendszernek. A másik eddig kiküszöbölhetetlennek bizonyult probléma, amikor a készülék pontosan észak felé mutat (0/360 fok). Ha ezen érték közelében van és átugorja ezt a küszöböt, akkor meg kell csinálnia majdnem egy egész kört a felhasználónak, ha a helyes irányba szeretne visszaállni. Az alkalmazás ilyen esetben olyan módon figyelmeztet, hogy hirtelen nagy frekvenciával rezgeti a készüléket. Olyan esetben, amikor hirtelen nagymértékű kitérést érzékel az alkalmazás, például meglökik a felhasználót, vagy valamilyen mágneses szenzort zavaró esemény bekövetkezik, akkor az alkalmazás egy hibajelző hangot ad és megállítja az egyenes vonalú haladás irányítását. Ha a hiba elhárult, vagy újra akarna indulni az előzőleg kiválasztott irányba, a felhasználónak csak újból el kell indítani az egyenes vonalú haladás irányítását, mert a hiba fellépésekor a kiválasztott irány értékét megjegyzi az alkalmazás.

Az alkalmazás beállítása vakok számára nehézkes lehet külső segítség nélkül. Ilyenkor javasolt egy látó felhasználó segítségét igénybe venni. Azonban többnyire egy vak személy az okostelefonját olyan alkalmazásokkal egészíti ki, amivel könnyebb tájékozódni a képernyőn. Az ilyen típusú alkalmazásokról már szó esett az 1. fejezetben. A tesztelés során kiderült, hogy többszöri gyakorlás után tudja a felhasználó elsajátítani és megérteni az alkalmazás működését. Ezért is alkalmaztam a leírás képernyőt, ahol az útmutató jellegű

szöveg elmagyarázza a felhasználónak, hogyan is kell működtetni az alkalmazást.

Véleményem szerint az alkalmazás a legjobban akkor lehetne használható, ha a felhasználónak nem kéne a kezében tartania a telefont mindvégig. A projekt továbbfejleszthető lehetne egy olyan megoldással, amivel a készülék egy nyakba akasztható szerkezetre lenne rácsatolva úgy, hogy a készüléket fixen, a helyes módon tartsa (előre irányban, álló módban és képernyővel felfele).

Irodalomjegyzék

[Ableson et al., 2011] W. Frank Ableson, Robi Sen, Chris King and C. Enrique Ortiz: Android in Action, Third Edition, Manning Publications, 2011

[Ekler et al., 2012] Ekler, P. – Fehér, M. – Forstner, B. – Kelényi, I.: Android-alapú szoftverfejlesztés – Az Android rendszer programozásának bemutatása, SZAK Kiadó, 2012, old.: 1-22, 309-310

[Lee, 2012] Wei-Meng Lee: Beginning Android 4 Application Development, Wrox Press, 2012

[McGill, 2013] Sabrina A. Panëels, Dylan Varenne, Je!rey R. Blum and Jeremy R. Cooperstock: The walking straight mobile application, ICAD 2013, old.: 25-32

[Mednieks et al., 2012] Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura: Programming Android, 2nd Edition, O'Reilly Media, 2012

[Meier, 2012] Reto Meier: Professional Android 4 Application Development, Wrox Press, 2012

[Milette et al., 2012] Greg Milette, Adam Stroud: Professional Android Sensor Programming, Wiley, 2012

[Pandur, 2012] Pandur Balázs: Érintőképernyős mobil alkalmazások vak és gyengénlátó felhasználók részére, Budapesti Műszaki és Gazdaságtudományi Egyetem, 2012

Online hivatkozások

[1] The walking straight mobile application, ICAD 2013: http://srl.mcgill.ca/publications/2013-ICAD-WS.pdf (2014.11.12.)

[2] Android verziók:

https://developer.android.com/about/dashboards/index.html (2014.11.19.)

[3] Mobile Accessibility: https://play.google.com/store/apps/details?id=es.codefactory.android.app.ma.vocalizerenu (2014.11.19.)

[4] Georgie: https://play.google.com/store/apps/details?id=net.screenreader.GeorgieLauncher (2014.11.19.)

[5] Jon Fingas (engadget.com): http://www.engadget.com/2014/01/29/strategy-analytics-2013-smartphone-share/ (2014.11.19.)

[6] Smartphone OS Market Share, Q2 2014 http://www.idc.com/prodserv/smartphone-os-market-share.jsp (2014.11.19.)

[7] Alliance FAQ. In: Open Handset Alliance http://www.openhandsetalliance.com/oha_faq.html (2014.11.19.)

[8] Android architecture: http://www.tutorialspoint.com/android/android_architecture.htm (2014.11.19.)

[9] Activity lifecycle: http://developer.android.com/reference/android/app/Activity.html (2014.11.19.)

[10] Android platform versions: https://developer.android.com/about/dashboards/index.html (2014. 11.19.)

[11] Sensors overview: http://developer.android.com/guide/topics/sensors/sensors_overview.html (2014.11.19.)

[12] Android SharedPreferences:

http://developer.android.com/training/basics/data-storage/shared-preferences.html (2014.11.19.)

[13] Sorkizárt szövegek készítése Androidon: https://github.com/bluejamesbond/TextJustify-Android/tree/8495997abb676dd86b7ee48d09a0bd6237adb5f4 (2014.11.19.)

[14] Straight Line Walk alkalmazás a Google Play-en: https://play.google.com/store/apps/details?id=com.johnny.straightlinewalk.app (2014.11.19.)

[15] AppBrain: Number of Android applications: http://www.appbrain.com/stats/number-of-android-apps (2014.11.19.)

Ábrajegyzék

Táblázatjegyzék