Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Pandur Balázs

Érintőképernyős mobil alkalmazások vak és gyengénlátó felhasználók részére

KONZULENSEK

Dr. Németh Géza

Tóth Bálint Pál

BUDAPEST, 2012

Tartalomjegyzék

Kivonat

Az elmúlt években a mobil készülékek, már nem csak telefonálásra, de egyéb kiterjesztett funkciók elérésére is képessé váltak, ezzel új paradigmát teremtve. A kibővített belső funkciók mellett a fizikai billentyűzetet szinte teljesen felcserélte az érintőképernyő, ami egy vak, vagy gyengénlátó felhasználó számára problémát jelenthet. További hátrányként jelentkezik, hogy az elérhető alkalmazások többsége sem megfelelő számukra, mivel a felhasználó általában nem kap elegendő hangos és egyéb típusú információt a készülék és az alkalmazások irányításához.

A ma piacon lévő népszerűbb mobil platformok közül mindössze az Apple támogatja különböző irányelvekkel az olyan alkalmazások fejlesztését, ami a látássérült emberek számára lehetővé tenné az érintőképernyős okostelefonok használatát, azonban ezeknek termékeknek magas ára szinte elérhetetlen néhány felhasználó számára. Célom egy olyan általános módszer kidolgozása volt, ami kényelmes és megbízható használatot garantál számukra és bármely érintőképernyős környezetben implementálható. Az ötlet alapját a Braille írás adta, ahol a látássérült ember az ujját domború betűkön húzva képes olvasni. Ez érintőképernyős környezetben is megvalósítható, azonban ebben az esetben tapintás alatt az egyes felhasználói vezérlők érintését kell érteni, amik beszéd alapú visszacsatolás formájában reagálnak a különböző felhasználói eseményekre.

A módszer hatékonyságának bemutatására egy operációs rendszer szinten beépülő beszélő billentyűzetet, továbbá egy SMS kezelő rendszert implementáltam. Az elkészült alkalmazások vakok és gyengénlátók számára optimalizált felhasználói felülettel rendelkeznek, mely egy általános TTS (Text-To-Speech) interfészt használ, és ezen keresztül közöl verbális információkat a felhasználó felé. A fejlesztéseket Google Android platformon végeztem, továbbá a beszélő menürendszerhez a BME-TMIT Beszédtechnológiai Laboratóriumában fejlesztett Profivox HMM (Hidden Markov-model) szövegfelolvasót használtam [1].

A dolgozatomban részletesen tárgyalom milyen problémákkal szembesülnek a vak és gyengénlátó felhasználók egy érintőképernyős készülék használatakor, áttekintem a korábbi rendszereket, bemutatom az általam megtervezett módszert, végezetül ismertetem az elkészült demonstrációs alkalmazásokat és a teszteredményeket.

Abstract

In recent years beside phone calls mobile devices were extended with other additional functions, thereby creating a new paradigm. Beside the extended functions, physical keyboard was almost replaced by touch screen, which could difficulties for a blind or visually impaired user. An additional drawback is that the most of the available applications are not appropriate for them, because they does not get enough information to control the applications and the device itself.

From the popular mobile platforms only Apple supports with different guidelines the development of applications, which could make using the touch screen easier for blind and vision impaired people, but due to the high prices of these products make inaccessible it in some cases. My goal was to design a general method which guarantees comfortable and reliable usage for them and it could realize any touch-screen environment. The basic idea came from the Braille writing, where visually impaired people are able to read by scanning their finger on the text. It could work also in a similar way in touch-screen environment, but scanning means here the touch of user controls, which respond with speech based feedback for the different user input events.

Beside working-out this technique, for demonstration I implemented two applications: (1) a virtual on-screen keyboard which can be integrated in the operating system, and (2) an SMS messaging system. These two solutions have user interface optimized for blind and visually impaired people and use general TTS (Text-To-Speech) interface which provide verbal information towards the user. The applications are Google Android based, and they are using the Profivox HMM (Hidden Markov-model) TTS engine, which was developed at the Speech Technology Laboratory of the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics (BME-TMIT) [1].

In this paper I discuss in details the main problems which affected the blind and vision impaired users on touch screen environment. Then I review the earlier available solutions, and introduce the designed method with the demonstration applications. Finally I examine the usability of these applications and I also introduce the test results.

1. Bevezetés

A 2001. évi népszámlálás adatai szerint Magyarországon az össznépesség 5,7 százaléka, azaz mintegy 577 ezer ember élt valamilyen fogyatékkal [2]. Az kimutatások szerint a gyengénlátó, vak, vagy az egyik szemére nem látó emberek száma a 3. legnagyobb arányban előforduló fogyatékossági forma hazánkban. A népszámlálás kutatást végzett a fogyatékkal élő lakosság iskolázottságáról is, ami kimutatta, hogy nagyobb arányban vannak azok, akik leérettségiztek, valamilyen szakmai oklevelet, vagy főiskolai, vagy egyetemi diplomát szereztek, mint azoké, akik még az általános iskola első osztályát sem végezték el. Ez részben annak is köszönhető, hogy létezik jó néhány olyan megoldás, ami lehetővé teszi a taníthatóságukat, segíthet a munkájukban és a mindennapi életüket is nagymértékben megkönnyítheti. Ilyen például a Braille írás, ami egy olyan írásrendszer, ahol a vak és gyengénlátó emberek tapintással képesek szöveget írni és olvasni [3]. Ezt a módszert 1824 óta használják és még ma is teljesen elfogadottnak tekintik, azonban a technika fejlődése ma már további megoldásokat is nyújt számukra. 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.

Napjainkban az érintőképernyők alkalmazása egyre elterjedtebbé válik. Az okostelefonok és a különböző hordozható eszközök mellett múzeumokban, repülőtereken és bevásárlóközpontban is használják őket. Ennek fő oka, hogy irányíthatóságuk bizonyos esetekben egyszerűbb és gyorsabb lehet, mint egy fizikai billentyűzettel ellátott készülék. A különböző interakciós technikák, mint például a forgatás, több ujjas gesztusok használata pedig még inkább népszerűvé tették ezt az interfészt [4]. Természetesen ez csak a látó felhasználók körében igaz. Számukra egy érintőképernyővel ellátott készülék jobb felhasználói élményt nyújt és valóban képesek azt gyorsabban és hatékonyabban kezelni, azonban ez egy vak vagy gyengénlátó ember számára megfelelő beszéd alapú visszacsatolás, vagy egyéb segítség nélkül nem lehetséges. Természetesen ebben a környezetben is léteznek különböző technikák (például képernyőolvasók), amik beszéd alapú visszacsatolással próbálják navigálni a felhasználót, azonban bizonyos esetekben ezek csak részleges megoldást nyújtanak.

Napjainkban az egyik legjobban elterjedt kommunikációs eszközök az okostelefonok. Ezek a készülék ma már nem csak hanghívásokra és SMS üzenetek küldésére használhatóak. A felhasználó elérheti az E-mail üzeneteit, olvashatja a friss híreket, vagy akár böngészheti a közösségi oldalakat is.

1. ábra Okostelefonok akkor és most

Az okostelefonok kialakítása nagy változáson ment keresztül a Symbian-os telefonok megjelenés óta (ld. 1. ábra). A régebbi készülékek még rendelkeztek fizikai billentyűzettel, azonban mára már ezek szinte teljesen eltűntek. A billentyűzet szerepét teljesen átvette az érintőképernyő, ill. az érintés érzékeny gombok használata, ami problémát jelenthet a vak, vagy gyengénlátó felhasználó számára. A fizikai billentyűzettel rendelkező készülékeken még volt lehetőség a nyomógombok kitapintására és így a készülék, illetve az alkalmazások irányítására is, azonban ezt pusztán az érintőképernyő használatával már nem tudják megtenni. A másik fő problémát a szoftverek kialakítása és irányíthatatlansága okozza. Ez főként annak köszönhető, hogy a mai mobil operációs rendszerek többsége nem támogatja semmilyen formában az olyan alkalmazások fejlesztését, amik megfelelőek lennének ennek a felhasználói körnek, továbbá a szoftverfejlesztők sem fordítanak erre különösebb figyelmet.

A dolgozatban egy olyan általam kidolgozott módszert ismertetek, ami érintőképernyős környezetben megfelelő megoldást nyújthat a látássérült emberek problémáira, illetve látó emberek számára is használható, továbbá platformtól függetlenül implementálható. A módszer kidolgozása előtt korábbi látóknak és vakoknak szánt megoldásokat vizsgáltam, amik előnyeit és hátrányait összegyűjtve definiáltam a szükséges funkciókat. Egy vak ember számára a kézenfekvő megoldást a képernyőolvasók alkalmazása jelentené. Ezek ma már érintőképernyős környezetben is léteznek, azonban egyik nagy hátrányuk, hogy nem egy alkalmazáshoz optimalizáltan olvassák fel a képernyőn látható komponenseket. Egyszerűbb alkalmazások esetében ez a megoldás is elegendő lehet, azonban komplexebb felhasználói felület esetén a képernyő-felolvasók nem képesek jól érthető módon feldolgozni a felhasználói vezérlők és egyéni nézetek sokaságát. Célszerű lenne tehát valamilyen más

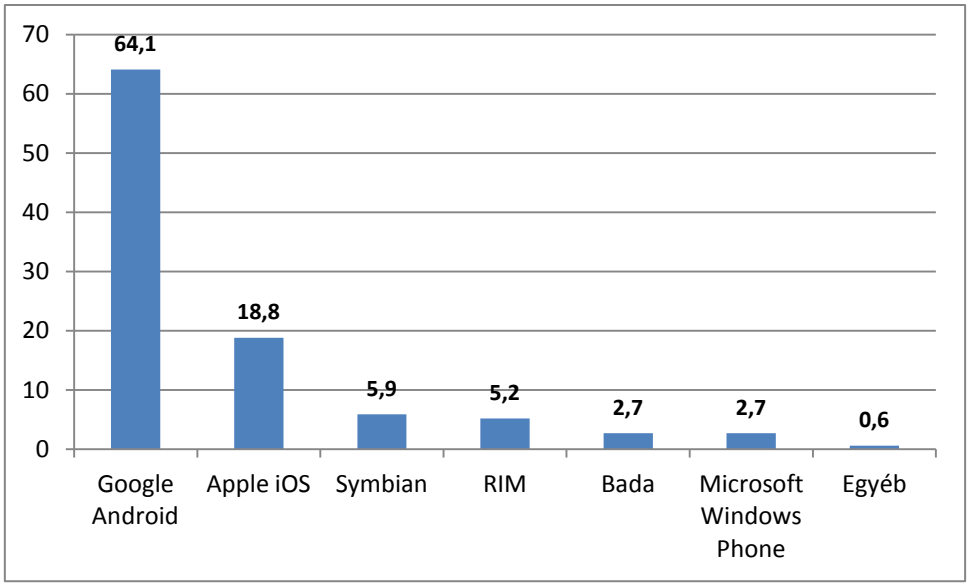
módszerrel informálni a felhasználót, például érintés érzékeny felhasználói vezérlők alkalmazásával, amik hang alapú visszacsatolás formájában segítik a navigációt a menüpontok között.

Az általam kidolgozott módszer hatékonyságának demonstrációjához két alkalmazást készítettem Google Android platformon. Mivel az SMS üzenetek írása az egyik legalapvetőbb funkció mobiltelefonos környezetben, azonban a vak és gyengénlátó felhasználók számára mégsem elérhető, ezért egy ilyen alkalmazás kifejlesztése mellett döntöttem. Az elkészült SMS kliens egy kiegészítéseként létrehoztam egy beszélő billentyűzetet is, ami pontosan ugyanezt a technikát alkalmazza, azonban további előnye, hogy operációs rendszer szinten beépül, így más alkalmazásokból is elérhető. A teszteket végül 5 látó és 1 vak felhasználó segítségével végeztem.

2. Mobil eszközök áttekintése

Az okostelefonok manapság az egyik legjobban elterjedt kommunikációs eszközök, amelyek ötvözik a PDA-k (Personal Digital Assistant) és a mobiltelefonok funkcióit. Ezek a készülékek az SMS és telefonhívások kezelésén kívül még számos egyéb kiegészítő funkcióval is rendelkeznek, továbbá fontos sajátosságuk, hogy saját operációs rendszert futtatnak, amik lehetőséget nyújtanak az alkalmazások fejlesztésére is. Az okostelefonok, már a 2000-es évek eleje óta, azaz közel egy évtizede jelen vannak a piacon. Nagy számban először a Nokia kezdett ilyen készülékeket gyártani, amelyek kezdetben Symbian operációs rendszert futattak. Már ezek a készülékek is tartalmazták a GPS, E-mail, Wi-Fi és hasonló funkciókat, azonban még nem rendelkeztek olyan erősségű hardverrel, mint mai társaik. Természetesen elmondható a mai modern készülékekről is, hogy szűkösebb erőforrásokkal rendelkeznek, mint egy laptop, vagy egy asztali számítógép, azonban ma már nem ritka, hogy több magos processzorral és több GB-os belső memóriával látják el ezeket a készülékeket.

A piacon operációs rendszer szinten 6 nagyobb mobil platformot említhetünk. Ezek a Google Android, Apple iOS, RIM (Research In Motion), Bada, Microsoft Windows Phone és a Symbian. Ezek piaci részesedése az eladott készülékek alapján a 2. ábrán látható.



2. ábra A mobil operációs rendszerek piaci arányai (% - ban) 2012 Q2-ben [5]

Az diagramon jól látszik, hogy a Google Android toronymagasan 64,1%-ban van jelen az eladott készüléken, majd ezt követi az Apple iOS-e 18,8%-al. A helyzetet, azaz az Android vezető pozícióját több dolog is indokolja. Fő előnye az Apple iOS-el szemben, hogy opensource, azaz egy nyílt forráskódú rendszer, amit sok gyártó előnyben részesít, mindemellett a Google Play-n rengeteg ingyenes, vagy olcsón elérhető alkalmazás áll a felhasználó rendelkezésére. Ezzel szemben az Apple operációs rendszere az iOS csak a saját készülékein képes futni, ami már önmagban indokolja miért szorult vissza eladások tekintetében a második helyre. Egy másik előnye a Google Androidnak, hogy ma már olcsóbban hozzá lehet férni az ilyen készülékekhez, míg egy iPhone esetében ez nem feltétlenül igaz.

A mai okostelefonok egyik legnagyobb előnye a programozhatóságuk. Minden operációs rendszer rendelkezik külön SDK-val (Software-Development-Kit), ami tartalmazza a platformhoz specifikusan tartozó osztálykönyvtárakat. Általában ezeknek a fejlesztőcsomagoknak minden évben megjelenik egy új változata, amiben szerepelnek az új és a javított funkciók is. Erre jó példa a Google Android 4.0-ás platformja az Ice Cream Sandwich, ami már képes táblagépen és okostelefonok is futni (korábban a táblagépekre való fejlesztése csak a 3.x.x verziószámú Honeycomb-al volt lehetséges). A mobil platformok fokozatos fejlődése és az okostelefonok népszerűsége fokozta a mérnökök fejlesztési kedvét is. Ma már minden mobil platformhoz létezik web áruház, ahonnan a felhasználók letölthetik a szükséges alkalmazásokat, a fejlesztők pedig közzétehetik termékeiket fizetős, vagy ingyenes formában. A felhasználó számára ma már nagy számban állnak rendelkezésére azok

az alkalmazások, amik a szórakoztatás mellett képesek a készülék funkcióinak kiterjesztésére is. Mindent összevetve elmondhatjuk tehát, hogy egy mai okostelefont nem is igazán a beépített funkciók megléte, hanem maguk a telepített alkalmazások teszik „okossá”. Ezt jól bizonyítja az is, hogy már az első készülékek is rendelkeztek a mai okostelefonok alapfunkcióival, azonban az elérhető szoftverek kis száma miatt ezek nem minden esetben voltak teljesen kihasználva.

3. Korábbi rendszerek vizsgálata

Általánosságban elmondható, hogy egy okostelefonra, vagy bármely más érintőképernyős környezetbe szánt alkalmazás nem megfelelően kezelhető egy vak, vagy gyengénlátó felhasználó számára. A fő problémát általában az jelenti, hogy a képernyő elrendezése olyan összetett elemeket tartalmaz, amik képernyő-felolvasó szoftver segítségével sem kezelhetőek. A probléma megléte azonban nem írható csak a fejlesztők számlájára, hiszen a legtöbb mobil operációs rendszer alapvetően nem támogatja rendszerszinten az olyan alkalmazások fejlesztését, ami egy vak, vagy gyengénlátó ember számára megfelelő lehetne. Egyedül az Apple rendelkezik olyan szolgáltatással, (VoiceOver), ami használható megoldást nyújthat, azonban a Google Android, vagy egyéb mobil platformok önmagukban ezt nem támogatják.

Egy mai okostelefon általában egy nagyméretű érintőképernyőből és mindösszesen néhány fizikai gombból áll, emellett a felhasználó általában különböző gesztusok segítségével képes navigálni a különböző menüpontok között. Ez az irányítási mód és az animált felhasználói felület egy látó ember számára remek felhasználói élményt nyújt, azonban egy vak, vagy gyengénlátó ember számára ez inkább hátrányként jelentkezik. Egy vak ember számára teljesen irreleváns a grafikus felhasználói felület kinézete, számára sokkal fontosabb az, hogy a különböző felhasználói vezérlők logikusan, és könnyen elérhetően legyenek elhelyezve a képernyőn. Léteznek természetesen különböző alkalmazások, amik segítséget nyújthatnak az érintőképernyő használatában, azonban általában ezek sajnos gyakran csak valamilyen részleges megoldást nyújtanak.

3.1 Slide Rule

A Slide Rule technika egy teljes mértékben vakoknak készült megoldás. A fejlesztők a felhasználók által legnépszerűbbnek tartott alkalmazásokat építették a rendszerbe, így például

elérhetőek a telefonkönyv, a zenelejátszó és az üzenetküldés funkciók is. Az alkalmazás nem rendelkezik hagyományos értelemben vett grafikus felhasználói felülettel, mindössze különböző színekkel jelzi a futási állapotát. A megjelenítés helyett a rendszer tervezői inkább az irányíthatóságra és a beszéd alapú visszacsatolás minél gondosabb tervezésére koncentráltak.

Hasonlóan egy általános alkalmazáshoz a felhasználó itt is gesztusok segítségével képes navigálni az egyes funkciók között. Annak ellenére, hogy semmilyen vizuális visszacsatolás nincs jelen, a rendszer mégis egy listát használ, ami a fel-le gesztusok segítségével léptethető, továbbá a jobbra-balra legyintésekkel érhető el az egyes listaelemekhez rendelt funkció. Ezzel a megoldással lista gyorsan és könnyedén irányítható, így például egy E-mail üzeneteket kezelő alkalmazást esetén, ha a felhasználó a továbbküldés funkciót szeretné elérni, akkor egy aktuális elemen jobbra húzással megteheti ezt, ahelyett, hogy a funkcióhoz hozzárendelt gombot keresgélné [6].

3.2 Mobile Accessibility

Ez az alkalmazás szintén egy kifejezetten vakoknak és gyengénlátó felhasználóknak szánt megoldás, ami több funkciót is magában foglal. Segítségével elérhetőek a névjegyzekék, SMS és E-mail üzenetek, az ébresztőóra, a naptár és egy böngésző is. A grafikus felhasználói felület mellett ez az alkalmazáscsomag beszédalapú menürendszerrel is rendelkezik, ami folyamatosan tájékoztatja a felhasználót. Az képernyőn látható felhasználói vezérlők hasonlóan működnek a Braille írásban használt írásjegyekhez. A komponensek érintésekor felolvasásra kerül a felhasználói vezérlő neve, funkciója, továbbá ha van, akkor a tartalma is (például szövegbeviteli mező esetében).

Fontos megemlíteni, hogy az alkalmazás fejlesztői több olyan összetett komponenst is beépítettek, ami egy vak, vagy egy gyengénlátó ember számára problémát okozhatna, azonban az interfész úgy lett kialakítva, hogy ezek mégis használhatóak legyenek. Alapesetben a problémát olyan felhasználói vezérlők alkalmazása jelentené, amelyek több nézetet jelenítenek meg. Ilyen például egy lista, vagy egy táblázatos megjelenítési forma, ahol az elemek szorosan egymás alatt helyezkednek el, és gyakran a felhasználó nem is látja az összes elemet a képernyőn (ld. 3. ábra). Az alkalmazás fejlesztői a problémát úgy oldották meg, hogy a listát soronként léptethetővé tették fel-le gesztusok segítségével, továbbá a képernyőn látható elemeket hasonlóan az egyszerű vezérlőkhöz érintés érzékeny módon definiálták.

3. ábra A Google Android alapértelmezett listás megjelenítése

A fenti képen egy listás megjelenítési forma látható. A Mobile Accessibility esetében a felhasználó az ujját húzva képes kiválasztani a képernyőn látható, neki megfelelő opciót, majd a képernyőn bárhol duplán kattintva képes megnyitni azt. A lista természetesen görgethető is. A felhasználó ezt az első, vagy az utolsó elemen állva egy fel-le gesztus segítségével teheti meg. Ez a megoldás lehetővé teszi az összetett vezérlők könnyű és biztonságos használatát, így például táblázatos nézetek esetén is alkalmazható.

Amellett, hogy ez az alkalmazás remekül használható rendelkezik néhány hátránnyal is. Ugyan kiválóan irányítható, viszont a menüpontok közötti navigáció lassú és néhány helyen hiányoznak belőle bizonyos funkciók is. Nagy problémát jelent továbbá a teljes alkalmazás magas ára is, ami szinte elérhetetlenné teszi [7].

3.3 VoiceOver

A VoiceOver az Apple által nyújtott szolgáltatás, ami lehetővé teszi vak és a gyengénlátó felhasználók számára, hogy iPhone készüléküket gond nélkül használhassák. Működése nagyon hasonló a Mobile Accessibility-hez. Itt is beszéd alapú visszacsatolás formájában érkezik információ a felhasználó felé és az irányítás is különböző gesztusokkal történik, azonban fő előnye az előbb említett rendszerhez képest, hogy operációs rendszer szinten támogatott, továbbá az Apple különböző fejlesztési irányelvekkel és eszközökkel is segíti a fejlesztőket, hogy az alkalmazások kompatibilisek legyen a VoiceOver rendszerrel [8].

3.4 Képernyő-felolvasók

Ezek az alkalmazások képesek operációs rendszer szinten beépülni és verbális információkkal ellátni a felhasználót. Általában elérhetőek minden mobil platformon, működésük minden esetben hasonló, céljuk a képernyő tartalmának minél részletesebb felolvasása. Ez a megoldás valóban jól használható, hiszen folyamatosan tájékoztatja a felhasználót, hogy éppen melyik alkalmazást nyitotta meg, lezárult-e a képernyő és így tovább, azonban sajnos nem minden esetben nyújt teljes körű megoldást. Vannak olyan felhasználói vezérlők, amiknél sajnos egyáltalán nem használható. Ilyen például egy listás nézet, ahol irányításkor a felhasználó nem kap semmilyen információt arról, hogy éppen melyik elemen tartja az ujját. A listaelemek felolvasása csak a Trackball használatával lehetséges, ami pedig csak néhány készüléken található meg (ld. 4. ábra).

4. ábra A Trackball ma már csak néhány készüléken található meg

Hasonló problémát okoz, ha a felhasználó egy letöltött alkalmazást szeretne irányítani. Ilyenkor előfordulhat, hogy a fejlesztő egyedi nézeteket hozott létre, amiket a képernyő-felolvasó nem képes értelmezni, vagy hasonlóan a listához csak a Trackball segítségével képes felolvastatni.

Elmondható tehát, hogy ezek a szoftverek jól használhatóak, rengeteg plusz információval látják el a felhasználót, azonban nem teljesítenek olyan jól, mint egy olyan alkalmazás, amihez jól optimalizált beszélő menürendszer is elérhető.

3.5 Szövegbeviteli lehetőségek

Az elrendezések mellett a másik nagy problémát az alapértelmezett beviteli lehetőségek jelentik, amik nehezen, vagy egyáltalán nem használhatóak egy vak, vagy gyengénlátó felhasználó számára. Az okostelefonokon ma már általában nem található fizikai billentyűzet. Helyette a szövegbevitelt egy virtuális billentyűzet segítségével oldották meg, ahol a felhasználó az érintőképernyőt használva képes szövegek bevitelére. Ez a megoldás viszonylag jól használható abban az esetben, ha a beépített virtuális billentyűzetet, valamilyen további kiegészítővel például képernyőolvasóval együtt használja felhasználó, azonban ilyenkor is csak a leütött karakterekről kap információt (ld. 5. ábra). Ilyenkor egy elütött karakternél a felhasználónak törölnie kell, ami lassúvá és körülményessé teszi a szövegek írását.

5. ábra Virtuális billentyűzet Google Androidon

3.5.1 Beszédfelismerés

Manapság a beszédfelismerés, vagy ASR (Automatic Speech Recognition) aktív kutatási téma. A BME-TMIT Beszédtechnológiai Laboratóriumában is készítettek már magyar nyelvű beszédfelismerőt [9], emellett létezik Google Androidra és Apple iOS-re fejlesztett diktáló szoftvert is [10]. A mai rendszereknek hátránya, hogy egy távoli adatbázist használnak a beszédminták dekódolására, így a használatukhoz szükséges internetkapcsolat, emellett a hosszabb mondatokat sem képesek pontosan felismerni. Ennek ellenére, azonban jól használhatóak és meggyorsíthatják a szövegbevitelt.

3.5.2 BlindType

A virtuális billentyűzet használatára egy másik megoldást jelenthet a BlindType elnevezésű operációs rendszer szinten beépülő virtuális billentyűzet [11]. Működése hasonló a

már sima telefonokból is ismert T9 beviteli lehetőséggel [12]. Lényege, hogy képes nagy hatékonysággal kijavítani a hibásan beírt szavakat. Képes teljesen értelmetlen karakterláncokból értelmes szavakat felismerni, míg a T9 beviteli megoldás gyakran csak 1-2 karakteres hibatűréssel működik. Ezt a megoldást is használhatják vakok, vagy gyengén látok, hiszen a virtuális billentyűzettel beírt szavak, automatikusan kijavításra kerülnek. Nagy hátránya, hogy a felhasználó nem kap beszéd alapú visszacsatolást, azonban valamilyen képernyőolvasó alkalmazással használva jó megoldást nyújthat.

3.5.3 SmartBraille

Ma már léteznek olyan érintőképernyőre optimalizált alkalmazások is, amik a Braille írást használják. Ilyen például a SmartBraille, ahol a felhasználó ezt a technikát alkalmazva vihet be szövegeket [13].

3.5.4 Bluetooth billentyűzet

Ezekhez a mobil készülékekhez ma már kapható Bluetooth-on keresztül csatlakozó billentyűzet is, ami segítséget nyújthat. Egy ilyen kiegészítő könnyebben irányíthatóvá teszi a készüléket, azonban hátránya, hogy csökkenti a mobilitást.

A vizsgált szoftveres megoldások előnyeit és hátrányait az alábbi táblázatban foglaltam össze:

	Slide Rule	Mobile Accessibility	VoiceOver	Képernyő olvasók	Smart Braille	Blind Type
Grafikus felhasználói felület	-	+	+	-	+	+
Beszéd alapú visszacsatolás	+	+	+	+	+	-
Magyar nyelv	-	-	+	+/-	-	-
OS szinten támogatott	-	-	+	+	+	+
Vak	+	+	+	+	+	-
Gyengénlátó	-	-	-	-	-	-
Látó	-	+	+	+	-	+

1. táblázat A vizsgált megoldások előnyei és hátrányai különböző szempontok szerint

A táblázat oszlopaiban a vizsgált megoldások neve olvasható, a sorokban pedig a vizsgált szempontok. Azoknál a szempontoknál, ahol az alkalmazás megfelelt az elvártaknak a cellát (+)-al jelöltem, ahol nem ott pedig (-)-t használtam. A képernyőolvasó „Magyar nyelv” mezőjét (+/-)-al jelöltem, ami mindössze azt jelzi, hogy önmagában egy ilyen

alkalmazás nem támogat semmilyen nyelvet, mindössze a telepített szövegfelolvasó szoftver segítségével képes megfelelően működni. A fenti táblázatban az is furcsa lehet, hogy a gyengénlátó felhasználók igényeit egyik alkalmazás sem teljesíti. Ennek oka, hogy a felsoroltak közül egyik sem teszi lehetővé, hogy a felhasználó különböző színsémákat állítson be, vagy skálázhassa a betűméreteket, holott ezeket a megoldásokat már régebb óta alkalmazzák különböző weblapok tervezésekor [14]. Természetesen ennek a felhasználói körnek is segítséget nyújthat a beszéd alapú visszacsatolás valamilyen formája, azonban ezt a megoldást nem minden esetben igénylik.

4. Érintőképernyőre optimalizált beszélő felhasználói felület

Az 3. fejezetben bemutattam néhány vakoknak és gyengénlátóknak szánt megoldást érintőképernyős környezetben. Ezek nem minden esetben nyújtanak teljes körű megoldást a problémáikra. Vannak ugyan jól használható rendszerek, mint például a képernyő-felolvasók, amik beszéd alapú visszacsatolás formájában tájékoztatják a felhasználót, azonban bizonyos esetekben ezek nem használhatóak, hiszen nem egy adott alkalmazáshoz optimalizáltan olvassák fel a képernyőn látottakat. Léteznek azonban olyan termékcsomagok is, mint például a Mobile Accessibility, ami több kifejezetten vakoknak és gyengénlátóknak szánt alkalmazást foglal magába, hátránya azonban, hogy magas ára miatt szinte elérhetetlen, hiányoznak belőle bizonyos funkciók és az irányítása is lassú.

Célom tehát egy olyan módszer kidolgozása volt, ami mind a vak, mind pedig a gyengénlátó emberek számára kényelmes és megbízható megoldást nyújthat. Az ötlet alapját a Braille írás adta, ahol a vak ember az ujját húzva képes olvasni. Érintőképernyő esetében ez ugyancsak működhet. Egy vak ember a megfelelő beszéd alapú visszacsatolás hiányában a képernyőn található komponensek elhelyezkedését próbálja először memorizálni, majd ez alapján irányítani az alkalmazást. Egy bonyolultabb, több menüpontból és opcióból álló szoftver esetében azonban ez nem lehetséges, emellett adódhatnak egyéb problémák is, amikkel egy vak, vagy gyengénlátó ember nem tud megbirkózni. Ilyenek például a görhető listás nézetek alkalmazása, amiket érintőképernyő környezetben gyakran használnak az alkalmazások fejlesztői, ugyanis egyszerű gesztusok segítségével könnyen irányíthatóak, továbbá átlátható megjelenítést tesznek lehetővé. Egy vak, vagy gyengénlátó felhasználó

számára azonban egy ilyen megjelenítés beszéd alapú visszacsatolás hiányában teljesen használhatatlan. Az ilyen problémák megoldására jól használható lenne, ha a képernyőn saját, érintés érzékeny felhasználói vezérlőket helyeznénk el, amik a beszélő menürendszeren keresztül elegendő információval látják el a felhasználót.

A fentiek alapján a felhasználó vezérlőket két osztályba lehet sorolni:

- egyszerű komponensek (gomb, szövegbeviteli mező, felirat, kijelölőnégyzet, és így tovább)
- összetett komponensek (listás, táblázatos megjelenítés, vagy valamilyen egyedi megjelenítési mód használata)

A fent említett felhasználói vezérlők első osztályába olyan komponensek sorolhatóak, amelyek nem tartalmaznak további nézeteket. Tervezési szempontból tekinthetjük ezeket könnyebben kezelhető elemeknek, hiszen mindössze arra van szükség, hogy valamilyen módon elérhetővé tegyük ezeket a felhasználó számára. Ekkor két megoldás lehetséges. Az egyik, hogy a képernyőn jól elkülönítve helyezzük el a különböző vezérlőket (például a képernyő négy sarkán), és a felhasználó számára felolvassuk az elérhető funkciókat az ablak megnyitásakor, a másik, hogy olyan érintés érzékeny felhasználói vezérlőket hozunk létre, amiket a felhasználó az ujja segítségével képes „kitapintani” a képernyőn. Kitapintás alatt természetesen nem a szokásos értelemben vett tapintást kell érteni, hanem valamilyen beszéd, vagy rezgés alapú visszacsatolást. Az utóbbi megoldásnak fő előnye, hogy lehetővé teszi a változatosabb megjelenítési formák létrehozását, azaz nem kell a különböző képernyőkomponenseket rögzített pozícióban elhelyezni, továbbá a felhasználónak nem kell azzal törődnie, hogy megjegyezze a különböző nézetek elrendezését, vagy újra és újra meghallgassa a képernyő felbukkanásakor elhangzó kezdeti információkat.

Ennél talán kicsit összetettebb megoldást igényel az olyan vezérlők alkalmazása, amik több komponenst is tartalmaznak. Tipikusan ezek a listás, vagy táblázatos megjelenítések, amik a különböző felhasználói vezérlőket egységbezárva kezelik. A fő probléma ezeknél a vezérlőknél az, hogy nem minden elem látszik a képernyőn, így nem lehetséges azok kitapintása sem, továbbá gesztusokkal lapozhatóak, ahol a felhasználó nem kap semmilyen információt az újonnan képernyőre került elemekről.

A probléma megoldására több módszert is kidolgoztam. Az eredeti megoldásokhoz hasonlóan ezek a módosított vezérlők is valamilyen gesztus segítségével irányíthatóak, azonban úgy, hogy vakok és gyengénlátó felhasználók is könnyedén használhassák.

Listás megjelenítés vezérlésére kidolgozott megoldások:

(1) Fel-le gesztus alkalmazása, elemek léptetése egyesével.

(2) Hasonlóan az (1) megoldáshoz a lista léptetéséhez fel-le gesztusok használata, azonban, ha a felhasználó lenyomva tartja az ujját a lista bizonyos késleltetéssel sorról-sorra ugrál fel, vagy le.

(3) A sorok tapinthatóak. Ha a felhasználó az ujját az első, vagy utolsó elemen hosszabb ideig nyomva tartja, akkor a lista egy oldalt görgethető.

(4) Hasonlóan a (3) megoldáshoz a lista sorai érinthetőek, azonban a görgetés fel-le gesztusok segítségével történik, emellett a felhasználónak lehetősége van a sorok egyenkénti léptetésére is a balról-jobbra, vagy jobbról-balra legyintések használatával.

A kidolgozott módszerek előnyeit és hátrányait az alábbi táblázat foglalja össze:

	Előny	Hátrány
(1)	Pontosan meghatározható a felhasználó számára, hogy éppen melyik soron áll	Lassú
(2)	(1)-es megoldáshoz képest gyorsabb irányíthatóság	Pontatlanság
(3)	A tapintásnak köszönhetően a sorok könnyebben elérhetek,	Az oldalak közötti lapozás lassú, és nem a megszokott módon történik
(4)	Kombinált módszer, ami a felhasználói igényekre szabhatóan irányítható	-

2. táblázat Composite, vagy összetett felhasználói vezérlők irányítására kidolgozott megoldások

A fenti módszerek tesztelésére mindegyik módszert implementáltam, majd a felhasználói észrevételek alapján javítottam őket. Már az első megoldás tökéletesen megfelelt volna a célnak, ha csak a vak, vagy gyengénlátó emberek igényeit tartottam volna szem előtt, azonban a látó felhasználók számára a sorok közötti navigáció lassú volt, továbbá nagy hátránya volt ennek a megoldásnak az is, hogy semmilyen lehetőség nem állt a felhasználó rendelkezése, hogy gyorsabban lépkedjen a sorok között. A probléma megoldását a második változat jelentette, ami értelemszerűen az első kibővített változata volt. Itt a felhasználó a listát automatikusan léptethette az ujját hosszabb ideig nyomva tartva. Ilyenkor lehetőség volt ugyan a gyorsabb léptetésre, azonban a felhasználó nem tudta pontosan detektálni a kívánt sor pozícióját, csak az átugrott sorok számáról kapott információt. A harmadik már sokkal

biztosabb hozzáférést nyújtott. A felhasználó ilyenkor már pontosan meg tudta határozni a sor helyzetét, hiszen hasonlóan egy sima felhasználói vezérlőhöz, képes volt a sorok kitapintására is. Itt a probléma a lapozásban volt, ugyanis a tesztelők idegennek és lassúnak találták ezt a módszert. A tényleges megoldást egy kombinált technika adta, ahol a felhasználó kitapinthatja a képernyőn látottakat, és mindemellett a fel-le gesztusokkal képes lapozni. A későbbi tesztek során felmerült az igény arra is, hogy a sorokat egyesével is lehessen léptetni, ezért a balról-jobbra, ill. a jobbról balra legyintések is bevezetésre kerültek.

A képernyőn látható komponensek pontos helyzetének meghatározása mellett egy vak, vagy gyengénlátó felhasználó számára minél egyszerűbb és megbízhatóbb irányítási technikát kell alkalmazni.

Az alkalmazott gesztusok/felhasználói események és funkcióik:

- *Koppintás:* felhasználói vezérlő kijelölése, felolvasása
- *Duplakattintás:* felhasználói vezérlő funkcióinak elérése
- *Jobbra/balra legyintés:* összetett felhasználói vezérlő esetén sorok egyenként léptetése
- *Fel/le gesztus:* összetett felhasználói vezérlő esetén lapozás a következő, vagy előző oldalra

Az alkalmazott gesztusok/felhasználói események külön felhasználói vezérlőkre bontva:

	Koppintás	**Dupla kattintás**	**Legyintés jobbra/balra**	**Fel/le gesztus**
Gomb	Funkció felolvasása	Funkció meghívása	-	-
Szövegbeviteli mező	Tartalom felolvasása	Billentyűzet megjelenítése	-	-
Címke	Felolvasás	-	-	-
Görgethető lista	Aktuális elem kiválasztása	A kiválasztott sorhoz rendelt funkció elérése	Sorok egyenkénti léptetése	Lapozás
Görgethető táblás megjelenítés	Aktuális elem kiválasztása	A kiválasztott elemhez rendelt funkció elérése	Táblázat elemeinek egyenkénti léptetése	Lapozás
Kijelölőnégyzet	Állapot felolvasása	Kijelölés, vagy kijelölés megszüntetése	-	-

3. táblázat Felhasználói vezérlők irányítása

A felhasználói vezérlőkhöz tervezett irányítási módszer, a duplakattintás bevezetése miatt elsőre lassúnak és körülményesnek tűnhet, azonban nagy előnye, hogy a felhasználó biztonsággal el tudja érni a kijelölt komponenshez rendelt funkciót. Fontos megjegyezni, hogy duplakattintás és a koppintás gesztusok egyike sem egy konkrét komponensen, hanem a képernyő bármely pontján elvégezhetőek. Ennek bevezetése azért volt szükséges, mert így a vak, vagy gyengénlátó felhasználó kisebb méretű komponensek esetén is képes egy adott elem funkciót elérni, míg sima koppintás esetén fennállhatna annak lehetősége, hogy mellébök, vagy egy másik elemre kattint rá.

5. Demonstrációs alkalmazások megtervezése

A 4. fejezetben kidolgozott módszer hatékonyságának bemutatására két demonstrációs alkalmazást hoztam létre. Mivel az SMS üzenetek kezelése még ma is az egyik legnépszerűbb üzenetkelése formának tekinthető mobiltelefonos környezetben, ezért egy olyan kliens fejlesztése mellett döntöttem, ami mind vak, mind látó, mind pedig gyengénlátó felhasználók számára lehetővé teszi ezeknek az üzeneteknek a kezelését [15]. Mivel ehhez elengedhetetlen egy olyan interfész, amin keresztül a felhasználó könnyedén tud szövegeket írni, ezért egy érintőképernyőre optimalizált operációs rendszer szinten beépülő beszélő billentyűzetet is létrehoztam.

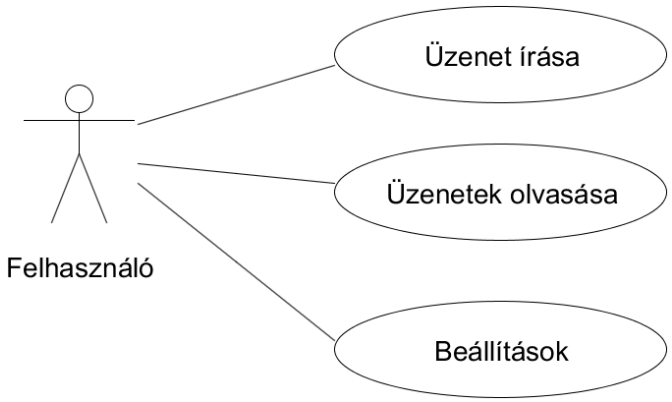
5.1 Szükséges szolgáltatások

Az SMS kliensnek minden olyan funkciót el kell tudni látnia, amit a felhasználó megszokhatott egy általános SMS üzeneteket kezelő alkalmazás esetében. A rendszernek tehát képesnek kell lennie az üzenetek fogadására, küldésére (akár csoportos formában is), névjegyzékek kezelésére, továbbá a beérkezett és elküldött üzenetek beszélgetésbe csoportosítására is.

Vak, vagy gyengénlátó felhasználó számára szükség van néhány plusz funkció beépítése is. Célszerű lehetőséget biztosítani a beszéd hangerejének, sebességének, vagy hangmagasságának beállítására is, továbbá látássérültek esetében különböző színsémák és betűméretek megválasztására. Ezek az igények felhasználónként változhatnak, ill. fontos megjegyezni azt is, hogy egyes gyengébb minőségű szövegfelolvasó szoftverek esetében az érthetőség ezektől a beállítási paraméterektől is függhet.

A megfelelő felhasználói felület mellett az alkalmazásnak olyan beszéd alapú intefésszel is rendelkeznie kell, ami elegendő információt nyújt az alkalmazás irányításához, azonban még nem zavaró a felhasználó számára.

Az SMS kliens által nyújtott szolgáltatásokat a következő leegyszerűsített UML2 Use-case diagram írja le:



6. ábra SMS kliens által nyújtott szolgáltatásokat ábrázoló UML2 Use-case diagram

A fenti Use-case-ek leírása:

Use-case neve	Üzenet írása
Rövid leírás	• Üzenet küldése (csoportos, vagy egyéni formában) • Beépített névjegyzék használata
Aktorok	Felhasználó

Use-case neve	Üzenetek olvasása
Rövid leírás	• Beszélgetésbe tömörítve, ami beszélgetésszerűen tartalmazza az elküldött és fogadott üzeneteket • Üzenetek törlése • Válaszüzenet küldése
Aktorok	Felhasználó

Use-case neve	Beállítások
Rövid leírás	• TTS (szövegfelolvasó) beállításai • Nyelvi beállítások • Színséma és betűméret választása
Aktorok	Felhasználó

5.2 Választott platform

Okostelefonokra ma már többféle platform létezik (ld. 2. fejezet). A legelterjedtebb mobil operációs rendszerek közé a Google Android és az Apple iOS sorolható, amik 64,1 és

18,8%-ban uralják a piacot. A fejlesztések megkezdésekor nem volt kérdés, hogy a két legnépszerűbb platform közül fogok választani, azonban mivel a Google Android jóval elterjedtebb és kedvezőbb árkategóriába esnek azok a készülékek, amik ezt a mobil operációs rendszert használják, ezért a választásom erre a platformra esett.

A Google Androidnak ma már számos verziója létezik 1.5-ös változattól egészen a 4.1-ig. A legelterjedtebb a 2.3.x verziószámú Gingerbread, ami a készülékek 55,8%-án található meg, azonban a 2.2-es változat 12,9%-val még mindig népszerűnek mondható [18]. Mivel a Google Android egy felfelé kompatibilis mobil operációs rendszer, továbbá a 2.2-es verziótól kezdődően a TTS (Text-to-Speech), és az ASR (Automatic Speech Recognition) funkciók már rendszerszinten támogatva vannak, ezért a választásom egyértelműen erre a platformra esett.

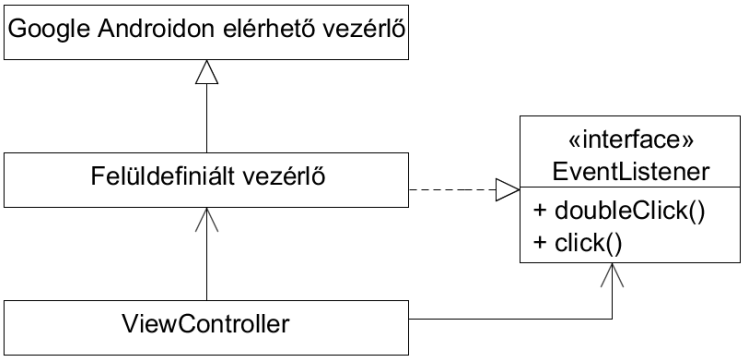
5.3 Kezdetben elérhető funkciók

Korábban már foglalkoztam hasonló témával. Akkor a cél egy SMS és E-mail üzenetek kezelő rendszer kidolgozása volt, ugyancsak vak és gyengénlátó felhasználók részére, azonban a megoldás teljesen más alapelven működött. A régebbi verzióban különböző alakzatok rajzolásával volt lehetősége a felhasználónak a különböző menüpontok és funkciók elérésére. A kezdeti fázisban ez ugyan jól működött, azonban később, a rendszer bonyolódásával az irányítás is egyre körülményesebbé vált, hiszen a felhasználónak többféle gesztust kellett megjegyeznie. A felmerült problémák miatt egy új metódus kidolgozása mellett döntöttem, ami akár komplexebb rendszerek esetében is könnyen használható.

5.4 Tervezés

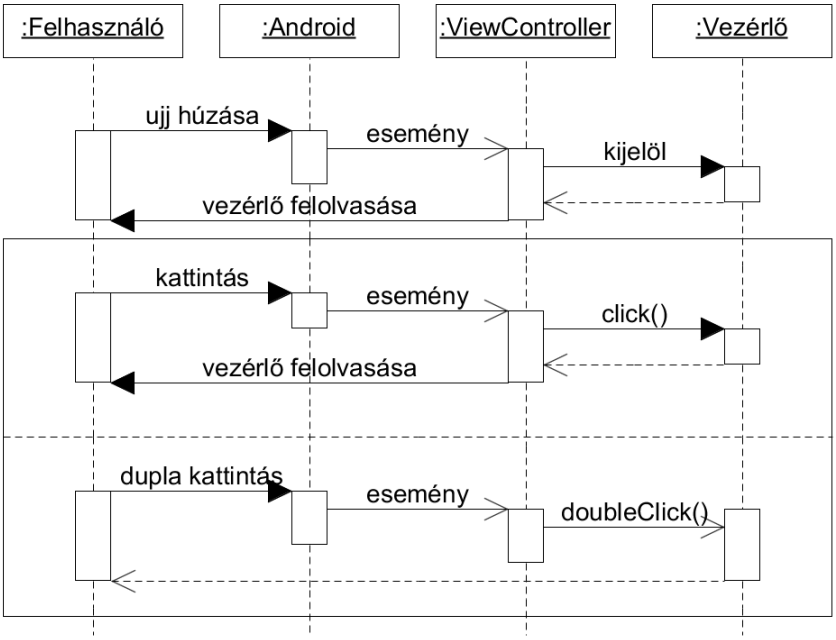
A demonstrációs alkalmazások fejlesztését az osztályok megtervezésével kezdtem [16][17]. A 4. fejezetben részletesen leírtam, hogy milyen követelményeknek kell eleget tennie egy vak, vagy gyengénlátó felhasználók számára készült alkalmazásnak érintőképernyős környezetben. Mivel azonban a Google Android fejlesztőcsomagja alapvetően nem támogatja az olyan vezérlők létrehozását, amik megfelelnének az említett felhasználói körnek, ezért ezeket magamnak kellett implementáljam. A demonstrációs alkalmazások fejlesztésekor összesen 8 különböző felhasználói vezérlőt definiáltam felül. Ezeket az SDK-ban megtalálható vezérlőkből származtattam, amik tartalmazzák az ősosztályaik szolgáltatásait, azonban a szükséges funkciókat felüldefiniálva, a felhasználói igényeknek megfelelőn alkalmazzák. Mivel ezeknek az új vezérlőknek többféle felhasználói

eseményre is megfelelően kell reagálniuk, ezért egy saját interfészt is létrehoztam, ami lehetővé teszi, hogy ezek megfelelően le legyenek kezelve. A hierarchiát a ViewController osztály kapcsolja össze, ami a különböző felhasználói eseményeket és a vezérlőkhöz tartozó funkciók meghívását vezérli (ld. 7. ábra).



7. ábra A felhasználói vezérlők kapcsolatát ábrázoló UML2 osztálydiagram

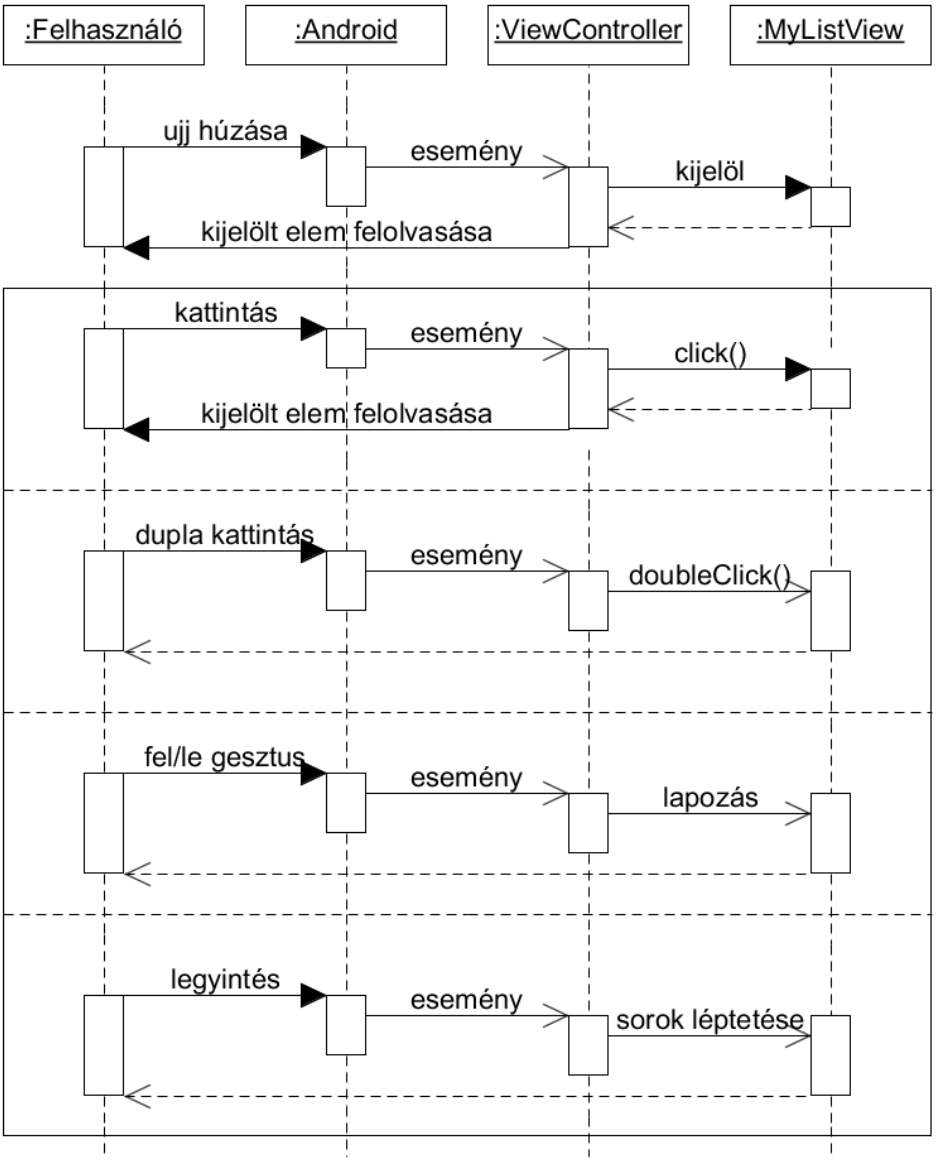
A felhasználó kezdetben az ujja húzásával tudja a vezérlőket „kitapintani”. Ekkor a ViewController osztály kijelöli a megfelelő vezérlőt, majd felolvassa annak nevét. A felhasználó ilyenkor koppintás segítségével újból felolvasathatja a kiválasztott komponenst, duplakattintással pedig elérheti a hozzátartozó funkciót (ld. 8. ábra).



8. ábra Felhasználói vezérlők működését ábrázoló UML2 szekvencia diagram

A fenti diagram nem vonatkozik az összetett (composite) vezérlők, például egy lista irányítására. Ilyen esetben egy bonyolultabb módszert kell alkalmazni, ahol a felhasználó

képes a kijelölt vezérlőt, a fel-le és jobbra-balra gesztusok segítségével irányítani, továbbá hasonlóan az egyszerű elemekhez az ujja húzásával egy sort, vagy elemet kiválasztani. Hasonlóan az Android SDK-ban található ListView-hoz, a fel-le gesztusok itt is a lista görgetését jelentik, azonban a felhasználó folyamatos beszéd alapú visszacsatolást kap a kiválasztott elemről, továbbá lehetősége van a listaelemek egyenkénti léptetésére is a jobbra-balra legyintések használatával (ld. 9. ábra).

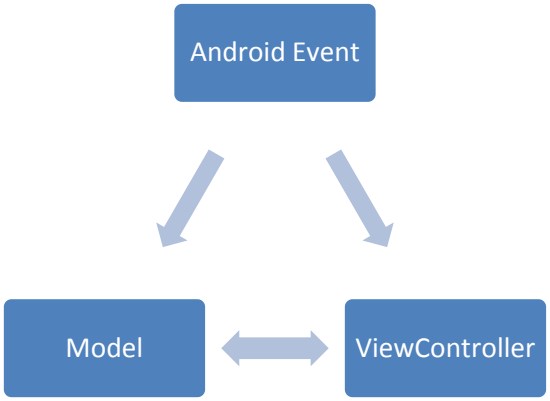


9. ábra Listás megjelenítés vezérlését ábrázoló UML2 szekvencia diagram

5.4.1 SMS kliens

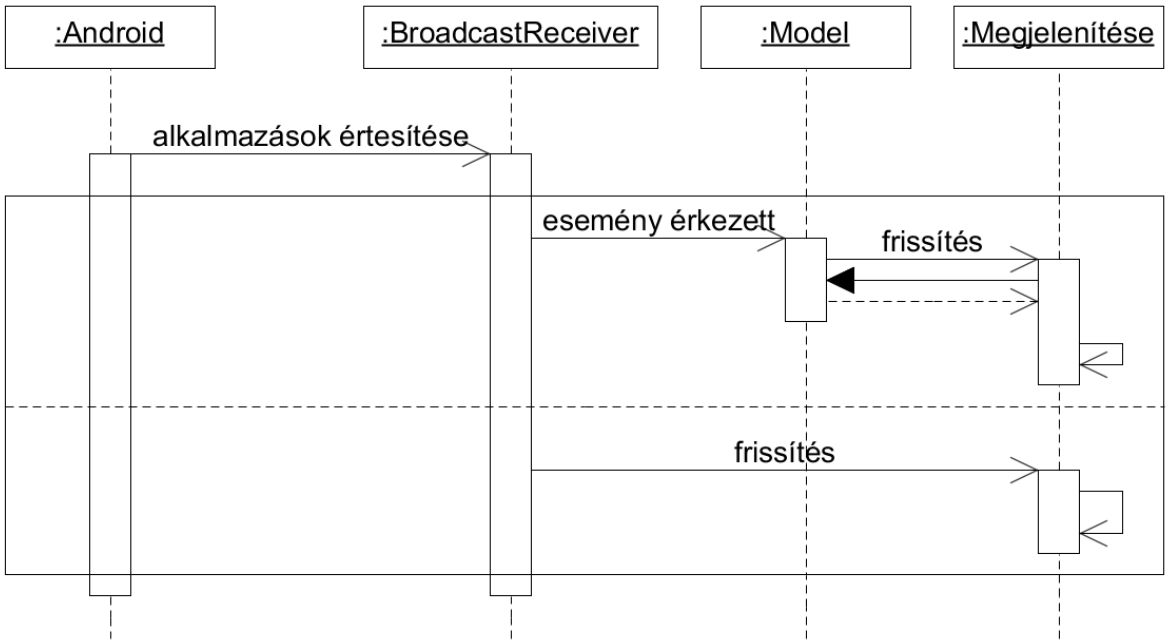
A rendszer tervezésekor és implementálásakor próbáltam minél nagyobb hangsúlyt fektetni arra, hogy a modell és a megjelenítés rész is jól elkülönüljenek egymástól. Ennek

érdekében az MVC (Model-View-Controller) tervezési mintát, ill. annak egy módosított változatát használtam (ld. 10. ábra) [19].



10. ábra MVC (Model-View-Controller) tervezési minta

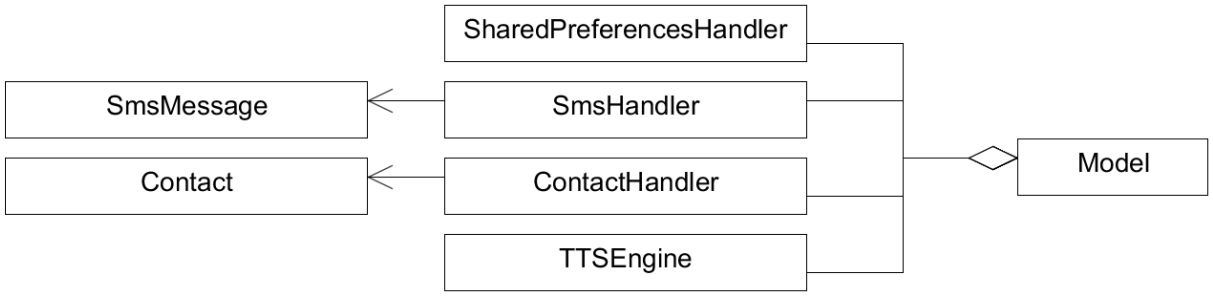
A Google Android alapvetően egy eseményvezérelt operációs rendszer, tehát egy esemény bekövetkezésekor (például egy SMS üzenet érkezése, képernyő elforgatása, és így tovább) a platform egy üzenetszórással értesíti a beregisztrált alkalmazásokat, jelen esetben az SMS klienst (ld. 11. ábra).



11. ábra Az operációs rendszer eseményeinek lekezelését ábrázoló UML2 szekvencia diagram

Ilyenkor az üzenetek elkapása a *BroadcastReceiver* osztály felüldefiniálásával tehető meg, ahol az esemény típusától függően az említett osztály meghívhatja a *Model* objektum szükséges metódusait, vagy közvetlenül frissítheti az aktuális nézet is.

A rendszer talán egyik legfontosabb eleme a *Model* osztály, ami az SMS kliens tényleges működéséért felelős komponensek vezérlését végzi (ld. 12. ábra)



12. ábra A főbb osztályok viszonyát ábrázoló UML2 osztálydiagram

Ez az osztály az alkalmazás indulásakor inicializálja a változóit, majd statikus elérést biztosít azokhoz.

Az SMS kliens funkcióit az alábbi főbb modulok valósítják meg:

- *SharedPreferencesHandler*: beállítások mentését/előhívását végző osztály
- *SmsHandler*: SMS üzenetek kezelését végző osztály
- *SmsMessage*: SMS üzenetet megvalósító osztály
- *ContactHandler*: névjegyzékek kezeléséért felelős osztály
- *Contact*: névjegyzéket megvalósító osztály
- *TTSEngine*: szöveg-beszéd átalakításért felelős osztály

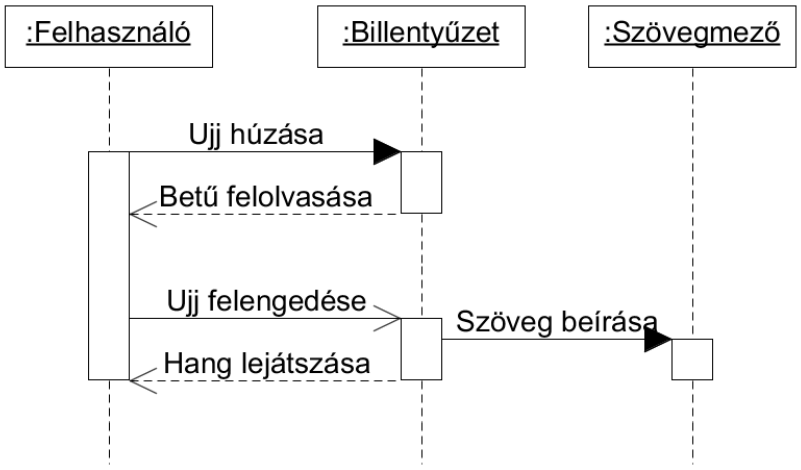
5.4.2 Beszélő billentyűzet

A beszélő billentyűzet fontos kiegészítője az elkészített SMS kezelő rendszernek, azonban még használhatóbb lenne, ha bármely más alkalmazásból képes lenne elérni ezt a felhasználó. Az Android fejlesztői csomagja lehetőséget biztosít úgynevezett Input Method Editor (IME) létrehozására, ami segítségével olyan szövegbeviteli eljárások fejleszthetőek, amelyekhez bármely alkalmazás képes hozzáférni [20].

A beszélő billentyűzet vezérlése hasonló elven működik az SMS klienséhez, azonban ebben az esetben szükség volt néhány apróbb módosításra is. A problémát a lassú irányíthatóság jelentette. Az eredeti tervek alapján a kiválasztott elem funkcióját a duplakattintás segítségével lehetett elérni, azonban a billentyűzet esetében ez nagyon lassúnak bizonyult. Sokkal kézenfekvőbb megoldást nyújt ebben az esetben, ha a duplakattintást felcseréljük az „ujj felengedése” eseménnyel. Ekkor a beszélő billentyűzet teljesen hasonló

módon képes működni, mint a beépített változat, azonban a beszéd alapú visszacsatolásnak köszönhetően a felhasználó számára rögtön felolvasásra is kerülnek a leütött karakterek.

A beszélő billentyűzet működését az alábbi szekvencia diagram szemlélteti:



13. ábra A beszélő billentyűzet működését ábrázoló UML2 szekvencia diagram

A felhasználó az ujja húzásával keresheti meg a neki megfelelő gombot, majd a képernyő felengedésével írhatja be azt. Ilyenkor az billentyűzet egy csipogó hang lejátszásával jelzi a betű beírását. A fenti ábra arra az esetre vonatkozik, amikor a felhasználó az ujját a billentyűzet területén belül engedi fel, különben a betű nem kerül beírásra. A karakterek beütésén kívül külön figyelmet kellett fordítani a szóköz és törlés gombok megvalósítására is. Ezek működése teljesen hasonló a sima karakterbillentyűkéhez, azonban fontos, hogy ezek leütésekor szóköz esetén az utolsó szó, törlés esetén pedig az utolsó szó(töredék) is felolvasásra kerül.

6. A megvalósított alkalmazások bemutatása

6.1 A beszélő menürendszer

A beszélő menürendszer (SUI-Speech User Interface) implementálásakor próbáltam minél nagyobb figyelmet fordítani arra, hogy a felhasználó minél több információt kapjon az alkalmazás használata közben, ügyelve azonban arra, hogy ez még ne legyen zavaró számára.

A beszélő menürendszer alapját a felüldefiniált felhasználói vezérlők adják, amelyek érintésre rezgés és hang alapú információ formájában tájékoztatják a felhasználót. A beszéd és rezgés alapú visszacsatolás mellett azonban fontos az is, hogy különböző hangok lejátszásával jelezzük egy nézeten történt esemény bekövetkezését (például új ablak megjelenését).

A felhasználó tájékoztatására valamilyen TTS (Text-to-Speech) motorra volt szükségem, ami képes a bemenetként kapott szöveget felolvasni. Egy ilyen szoftvernek az egyik legnagyobb előnye, hogy nem szükséges előre rögzített hangmintákat tárolni a telefon memóriájában, vagy egy távoli adatbázisban, mert a használt szövegfelolvasó motor végzi el a szükséges szintézist, így gyengébb hardverrel rendelkező készülékek esetében is használható.

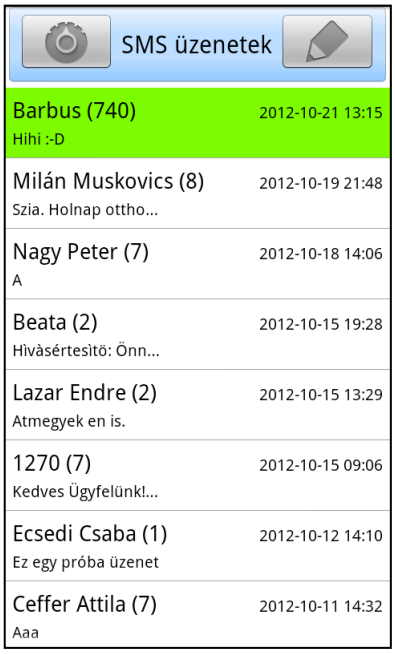
A rendszer fejlesztésekor a Beszédtechnológiai Laboratórium által fejlesztett Profivox HMM-et és a Google Play-n is megtalálható angol nyelvű Pico TTS-t használtam. A Profivox HMM egy rejtett Markov modell alapú szövegfelolvasó. Előnye, hogy kis adatbázisból is képes jó minőségű beszédet előállítani (1,5-2 Mbyte), továbbá rövid hangminták segítségével tanítható is, ami elméletileg lehetővé teszi a beszéd stílusának és karakterisztikájának javítását. Fontos további előnye, hogy magyar nyelvű, aminek a segítségével az angolt nem beszélő felhasználók is képesek lesznek az alkalmazás használatára.

6.2 Az SMS kliens megvalósítása

Az üzenetkezelő rendszer grafikus felhasználói felületét próbáltam úgy kialakítani, hogy vak és gyengénlátó felhasználhatók számára is használható legyen, és mindemellett jó felhasználói élményt nyújtson a látó emberek számára is. Az elkészült alkalmazás tehát nem csak a látássérültek, de az olyan látó emberek számára is hasznos lehet, akik autóvezetés közben szeretnének üzenetek küldeni, vagy olvasni.

6.2.1 Főmenü

Az különböző ablakok létrehozásakor próbáltam törekedni a minél egyszerűbb megjelenítésre. Így van ez a főmenü esetében is (ld. 14. ábra).



14. ábra A főmenüről készült képernyőkép

A fenti képernyőképen az elkészült SMS kliens főmenüje látható. Látszólag teljesen hasonlónak tűnik egy általános látóknak szánt alkalmazáshoz, azonban a beszélő menürendszeren és a különböző beállítási lehetőségeken keresztül a vak és gyengénlátó felhasználók számára is használható megoldást nyújthat.

A főmenüből elérhető funkciók a következők:

- Üzenet írása
- Beállítások menü
- Beérkezett és elküldött üzenetek olvasása beszélgetésekbe tömörítve

6.2.2 Üzenet írása menüpont

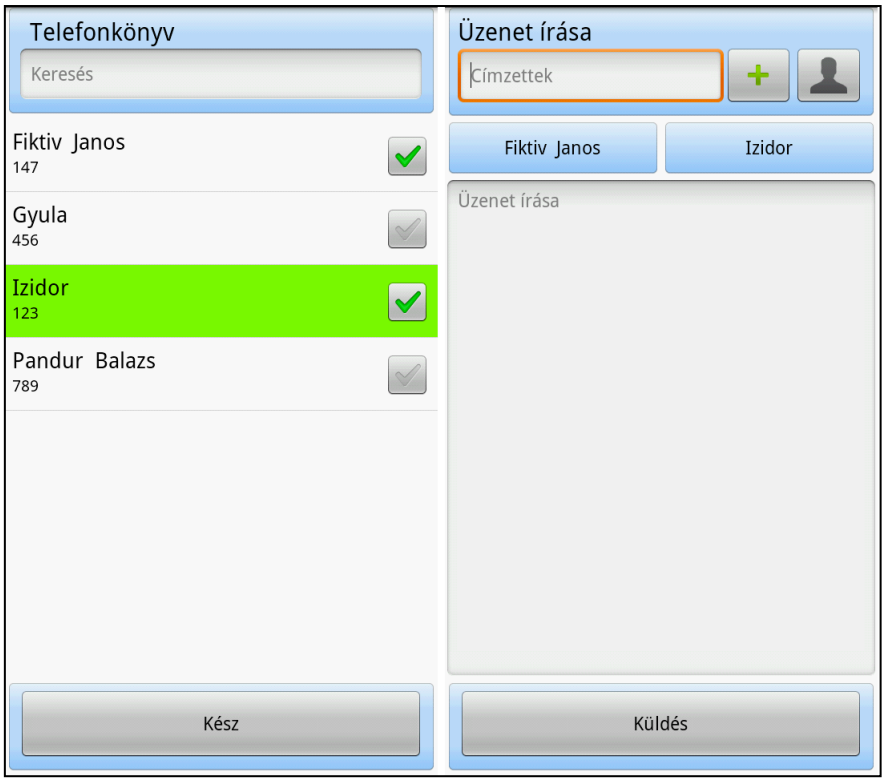
Az SMS kliens alapvető szolgáltatásként kell tartalmazza az üzenetküldés funkciót. A felhasználónak ebben a nézetben lehetősége van csoportos, vagy egyéni üzenetek küldésére is, amiket a rendszer a kézbesítés után a megfelelő adatbázis táblába ment, így a továbbiakban ezek a többi SMS kezelő rendszer számára is elérhetőek lesznek.

Egy Google Android operációs rendszert futtató készüléken a felhasználó adatai, mint például a telefonkönyv, hívásnapló, vagy SMS üzenetek egy köztes, minden alkalmazás számára elérhető adatbázisba kerülnek mentésre. Ezek a *ContentResolver* osztály segítségével frissíthetőek, módosíthatók, vagy törölhetőek is (természetesen ehhez az alkalmazás számára engedélyezni kell a hozzáférést). A kézbesítés után az elküldött üzenetek is ebbe az adatbázistáblába kerülnek mentésre.

Az üzenet írása menü a következő képen látható:

15. ábra Az üzenet írása menüpontról készült képernyőkép

A fenti ábráról a kék mezőben található vezérlők funkcióit emelném ki. A felhasználó itt több lehetőség közül is választhat. Használhatja a telefonkönyvet (legszélső gomb), és/vagy maga adhatja hozzá a címzetteket (zöld pluszjellel ellátott gomb). Utóbbi esetben a címzettek beírása manuálisan történik telefonszám, vagy a névjegyzékben szereplő bejegyzés neve alapján. Ha a felhasználó érvénytelen telefonszámot, vagy egy névjegyzékben nem szereplő bejegyzés nevét adta meg, akkor az alkalmazás ezt egy felbukkanó ablak segítségével jelzi. További opcióként lehetőség van csoportos üzenetek írására is. A felhasználó ezt megteheti, ha a címzetteket egyesével, a telefonkönyvön keresztül, vagy ha pontosvesszővel elválasztva adja meg (ld. 16. ábra).



16. ábra Az alkalmazás telefonkönyvéről és két hozzáadott névjegyzékről készült képernyőkép

A fenti ábra baloldalán az alkalmazás telefonkönyve látható. Ebben a menüpontban a felhasználónak lehetősége van keresni a már meglévő névjegyzékek között, továbbá szerkesztheti az üzenet címzettjeinek listáját is. A képernyőkép jobboldalán az üzenetek írása menüpont látható két hozzáadott névjegyzékkel, amelyek gombok formájában jelennek meg a felhasználói felületen. Ez a megoldás könnyebbé teszi a hozzáadott címzettek listájának kezelését, ugyanis azok már, mint kitapintható komponensek jelennek meg a képernyőn. A címzettek listája módosítható, a telefonkönyv segítségével, és/vagy a hozzáadott név/telefonszám törlésével (kitapintás, majd törlés a duplakattintás segítségével).

6.2.3 Beállítások menü

Ez a menüpont lehetőséges biztosít a felhasználó számára, hogy egyénre szabhassa a használt szövegfelolvasó motor beállításait, az üzenetkezelő rendszer nyelvét, továbbá kiválaszthassa a neki legjobban megfelelő színsémát és a hozzátartozó betűméretet is.

Egy vakoknak és gyengénlátóknak szánt alkalmazás esetében ezek az opciók különös odafigyelést igényelnek. A vak felhasználók csak a beszélő menürendszeren keresztül kapnak információt a képernyőn történtekről, tehát esetükben szükség lehet a beszédtempó, a hangmagasság és a hangerő pontos beállítására is. Egy gyengénlátó felhasználó azonban nem

minden esetben igényli ezt, számára bizonyos esetekben elegendő a különböző színsémák és betűméretek alkalmazása.

A Google Android lehetőséget biztosít a fejlesztők számára, hogy különböző stílusokat hozzanak létre. Ezt kihasználva az egyes színprofilokat és magukat a képernyőn megjelenő elrendezéseket is XML (Extensible Markup Language) fájlokban definiáltam.

Az alábbi kódrészlet az alapértelmezett színséma definícióját mutatja:

```
<style name="Theme.Default" parent="android:Theme.Light">
  <item name="titleLayout">@style/defaultTitleLayout</item>
  <item name="imageButton">@style/defaultImageButton</item>
  <item name="activityBackground">@style/defaultActivityBackground</item>
  <item name="bubbleLeft">@drawable/gray_left</item>
  <item name="bubbleRight">@drawable/gray_right</item>
  <item name="bubbleLeftFocused">@drawable/green_left</item>
  <item name="bubbleRightFocused">@drawable/green_right</item>
  <item name="listRowDefault">@color/white</item>
  <item name="listRowSelected">@color/pressed</item>
</style>
```

A kódrészletből látható, hogy egy színprofil több attribútumból áll. Ezeket az egyes felhasználói vezérlőkhöz a *style=”?paraméternév”* segítségével lehet hozzárendelni:

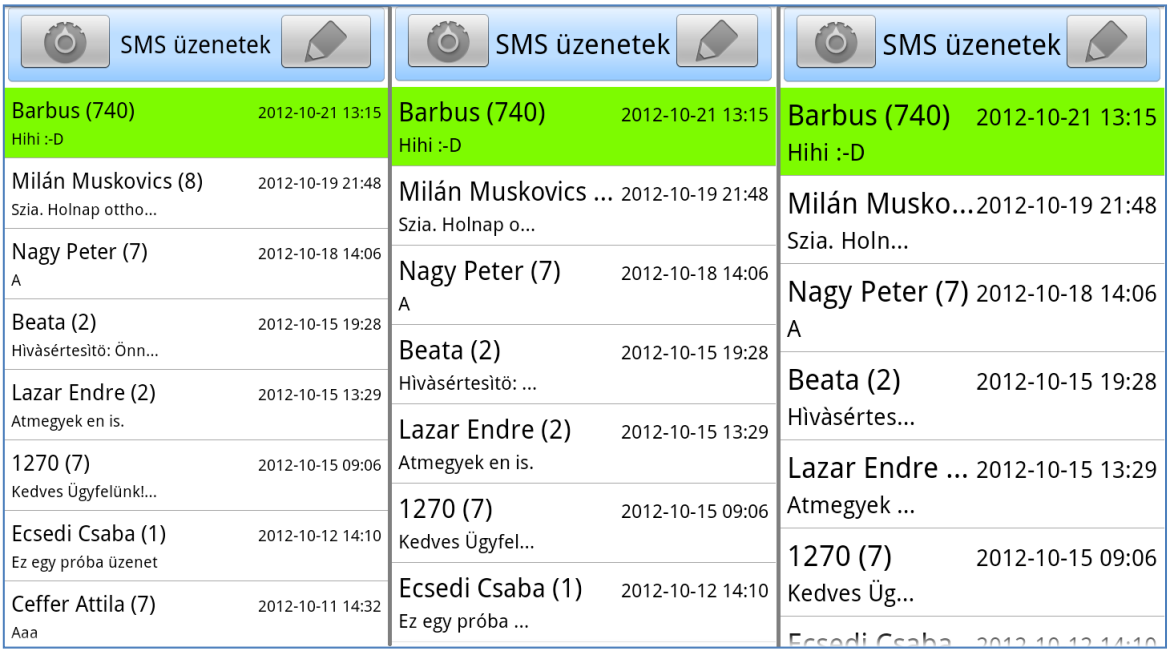
```
<com.blind.components.MyImageButton
  android:id="@+id/compose"
  style="?imageButton"
  android:layout_width="wrap_content"
  android:layout_height="50dip"
  android:layout_weight="0.5"
  android:src="@android:drawable/ic_menu_edit" />
```

A fenti megoldás segítségével az alábbi színsémákat hoztam létre normál, nagy és hatalmas méretekben:

- Alapértelmezett
- Fehér alapon fekete betűk
- Fekete alapon fehér betűk
- Sárga alapon fekete betűk
- Fekete alapon sárga betűk

A választás azért esett ezekre a színprofilokra, mert általában ezek azok, amiket más gyengénlátóknak szánt rendszereknél is használnak, továbbá különböző weblapoknál is alkalmaznak. A létrehozott színsémákról és a különböző méretekről készített képernyőképek a 17-18. ábrán láthatóak.

17. ábra A gyengénlátók számára készített színprofilok képernyőképei

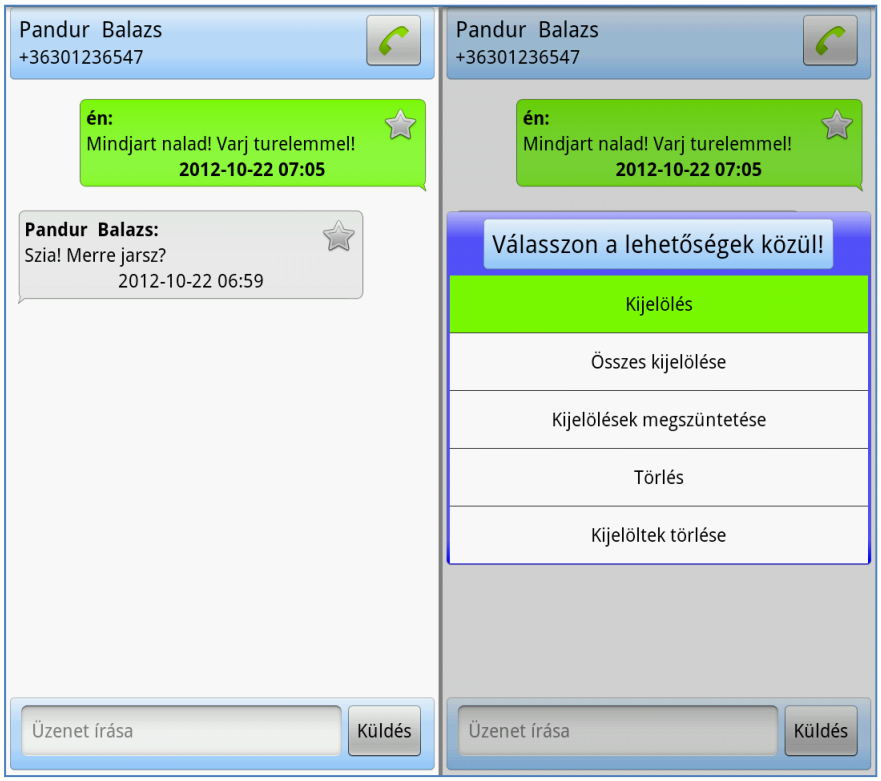


18. ábra A különböző méretprofilok az alapértelmezet téma esetén

Az elkészült alkalmazás további beállítási funkciójaként szerepel, hogy képes több nyelv támogatására is. Az alkalmazás egyelőre csak az angol és magyar nyelveket használja, azonban mind a tervezéskor, mind az implementáció során ügyeltem arra, hogy ez könnyen bővíthető legyen. Gondolva arra, hogy a rendszer később több nyelvvel is kibővíthető legyen, a felhasználói vezérlők különböző feliratait és a szövegfelolvasó által használ szövegeket is külön XML fájlokban tároltam.

6.2.4 Üzenetek olvasása

Ebben a menüpontban a felhasználó egy beszélgetéshez tartozó üzeneteit kezelheti. Beszélgetés alatt itt üzenetek olyan csoportját értem, amik ugyanattól a személytől érkeztek, vagy lettek címezve. Az üzenetek egymás alatt helyezkednek el a küldés és a fogadás időpontjának sorrendjében egy listában megjelenítve. Az üzenetek olvasása mellett felhasználónak lehetősége van továbbá válaszüzenet írására, a másik fél felhívására, vagy az üzenetek törlésére is. Az opciók menüpont egy üzenet kiválasztásával, majd duplakattintással érthető el (ld. 19. ábra).



19. ábra A beszélgetéseket megjelenítő nézetek

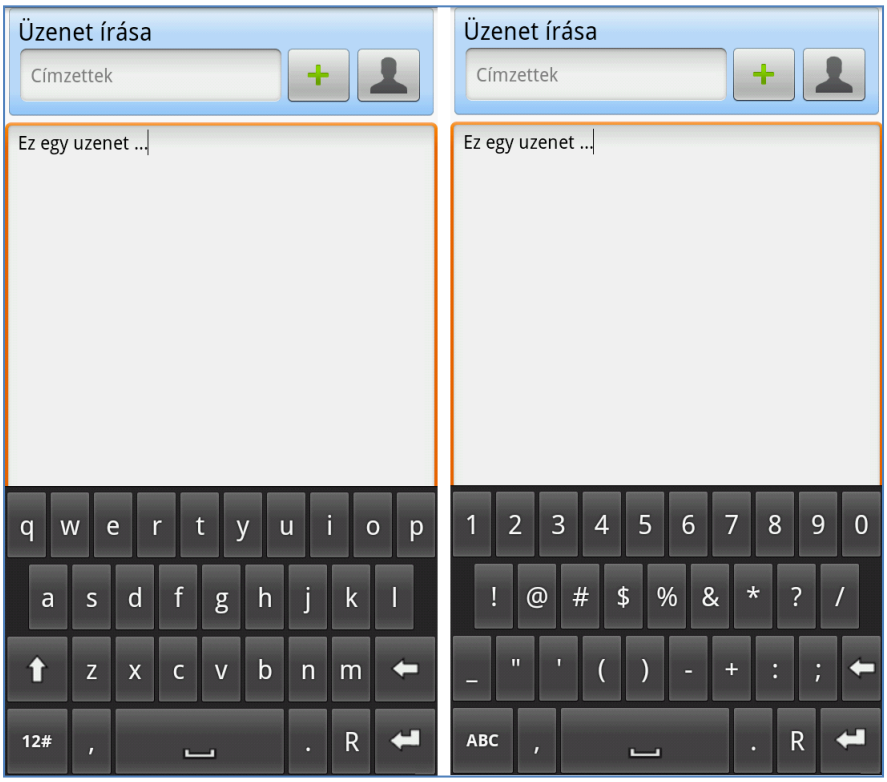
6.3 A beszélő billentyűzet megvalósítása

A kifejlesztett beszélő billentyűzet teljesen megszokott módon működik a Google Androidba beépített változathoz, azonban bármilyen képernyőolvasó nélkül könnyedén használható vakok és gyengénlátó felhasználók számára is (ld. 20. ábra). Előnye, hogy irányítása szinte teljesen azonos az SMS kezelő rendszerével és más alkalmazásból is elérhető.

Amellett, hogy a billentyűzetet érintés érzékeny gombokból építettem fel, a beszélő felületbe további funkciókat is implementálnom kellett, amik plusz információval látják el a felhasználót egy szöveg gépelésekor.

A fent említett funkciók a következőek:

- Felolvasás gomb (a billentyűzeten „R”-el jelöltem)
- Szóköz beütése után az előző szó felolvasása
- Törlés gomb megnyomásakor az utolsó szó(töredék) felolvasása



20. ábra Az operációs rendszer szinten beépülő beszélő billentyűzet

7. Tesztek, eredmények

A tesztelés fontos része egy alkalmazás fejlesztésekor. Ez így volt az SMS kezelő rendszer és a beszélő billentyűzet esetében is. A felhasználói tesztek mellett fontos a moduláris és különböző stressz tesztek elvégzése, hogy az alkalmazás a későbbiekben is az elvárt igényeknek megfelelően működjön.

Mivel a Google Android egy nyílt forráskódú operációs rendszer ezért több gyártó is előszeretettel használja ezt a platformot. A kijelző mérete, felbontása, a készülék processzora, belső memóriakapacitása gyakran eltér egymástól. Célszerű tehát az alkalmazást több készüléken is tesztelni.

Az alkalmazást a következő készülékeken teszteltem:

Készülék	Android verziószám
Sony Ericsson Xperia X10 mini pro	2.3.7
HTC Desire	2.3.3
HTC Wildfire S	2.3.5
Virtual Device	2.2

4. táblázat A tesztelésnél használt készülékek

7.1 Előkészületek a felhasználói tesztek előtt

7.1.1 Alkalmazások előkészítése

Az olyan alkalmazások esetében, amik még tesztelési fázisban vannak, gyakran a moduláris tesztek elvégzése után is előfordul, hogy a fejlesztők nem kezeltek le bizonyos eseteket és az alkalmazás valamilyen kivétel dobásával leáll. A tesztelők azonban nem ismerik az alkalmazás belső működését, így a probléma okát gyakran nem is tudják pontosan azonosítani. Megoldást jelenthet, ha valamilyen módon a fejlesztő elmenti az előforduló hibaüzeneteket egy log fájlba (tipikusan az SD kártyára), majd a tesztelési időszak után lementi azt a készülékről.

Google Android esetében ez a következő kódrészlettel tehető meg:

```
Thread.setDefaultUncaughtExceptionHandler(new UncaughtExceptionHandler() {
      public void uncaughtException(Thread thread, Throwable ex) {
            // Hiba fájlba írása
      }
});
```

7.1.2 Monkey Tool

Az Android Monkey Tool lehetőséget nyújt alkalmazások úgynevezett stressz tesztelésére [21]. Lényege, hogy a grafikus felhasználói felülettel rendelkező alkalmazások számára megadott számú, véletlenszerű felhasználói eseményeket generál (különböző gesztusok, kattintás, hangerőgombok megnyomásának szimulálása, és így tovább). Fontos előnye továbbá, hogy használható emulátoron és készüléken egyaránt.

7.1.3 Tesztelés felhasználókkal

Az alkalmazásokat összesen 5 látó és 1 vak felhasználó segítségével teszteltem. A látó emberek közül 3 férfi és 2 nő, a legfiatalabb 19 a legidősebb pedig 53 éves volt. Ezek a tesztelők a legtöbbet a grafikus felhasználói felület kinézetének és a funkciók javításában tudtak segíteni. A tesztelés legfontosabb része azonban az volt, amikor találkoztam Ecsedi Csabával egy vak programozó matematikussal, aki az Óbudai Egyetemen vesz részt látássérültek informatikai oktatásában. Csabával összesen idáig kétszer találkoztam, azonban sok olyan hiányosságra rávilágított, amire egy látó ember nem gondolt volna. Problémaként merült fel például a különböző hangjelzések használatának hiánya, vagy listás megjelenítések esetében az irányíthatóság pontatlansága/lassúsága. Véleménye szerint azonban a hibák ellenére a két demonstrációs alkalmazás és maga a módszer is jó kezdeményezés, és kisebb módosításokkal használható megoldást nyújthat a látássérültek problémáira érintőképernyős környezetben. A legutóbbi találkozásunkkor megbeszéltük, hogy a javaslatai alapján javítom az alkalmazásokat, majd az új változatot eljuttatom hozzá.

Összehasonlításképpen az elkészült rendszer funkcióit az alábbi táblázat foglalja össze:

	Slide Rule	**Mobile Accessibility**	**Voice Over**	**Képernyő olvasók**	**Smart Braille**	**Blind Type**	**Saját**
Grafikus felhasználói felület	-	+	+	-	+	+	+
Beszéd alapú visszacsatolás	+	+	+	+	+	-	+
Magyar nyelv támogatása	-	-	+	+/-	-	-	+
OS szinten támogatott	-	-	+	+	+	+	+
Vak	+	+	+	+	+	-	+
Gyengénlátó	-	-	-	-	-	-	+
Látó	-	+	+	+	-	+	+

5. táblázat A kifejlesztett rendszer funkciói a többi vizsgált módszerrel szemben

8. Jövőbeli tervek

A jövőben szeretném a módszert tovább tökéletesíteni és javítani a már meglévő rendszerek funkcióit. Mivel a szövegfelolvasók nem tartalmaznak semmilyen ékezetesítő motort, ezért ez a bizonyos esetekben a felolvasás rovására mehet. A probléma megoldására szeretnék egy ékezetesítőt integrálni a rendszerbe, ami egyfajta preprocesszorként működne.

Egy másik problémát jelent, ha a felhasználó több nyelven kap üzeneteket. Ekkor a beállított TTS motor nem képes megfelelően felolvasni azt. A felhasználónak ilyenkor meg kell változtatni a beállított szövegfelolvasót, ami kényelmetlenné teheti a rendszer használatát. A probléma orvosolható lenne nyelvdetekció használatával, majd a felismert nyelvnek megfelelően a szövegfelolvasó motor dinamikus változtatásával.

Kitűzött célként a kifejlesztett módszerrel szeretnék több hasonló alkalmazást implementálni, emellé pedig egy olyan termékcsomagot létrehozni, ami a vakok és gyengénlátó felhasználók által leggyakrabban használt funkciókat tartalmazza.

9. Összefoglalás

A mai mobil operációs rendszerek egyike sem tartalmaz rendszerszinten olyan funkciókat, ami közvetlenül támogatná a vak és gyengénlátó felhasználók igényeit. Léteznek ugyan látássérültek számára készült rendszerek, mint például a Mobile Accessibility, vagy a különböző képernyőolvasók, azonban ezek mindegyike rendelkezik valamilyen hátránnyal, ami miatt csak részleges megoldásnak tekinthető. Mivel a kutatások során egy olyan rendszert sem találtam, ami árban elérhető, továbbá minden felhasználói igénynek eleget tenne, ezért egy olyan módszer kidolgozása mellett döntöttem, ami a vizsgált rendszerek előnyeit kombinálva, plusz azok hátrányait, hiányosságait javítva egyszerű és kényelmes megoldást nyújthat a látássérültek problémáira egy érintőképernyős készülék használatakor. A módszer alapját egy Braille íráshoz hasonló technika adta, ahol a felhasználó az ujja húzásával volt képes a különböző képernyőn található komponensek kitapintására, amikor is a rendszer hang és rezgés alapú visszacsatolás formájában tájékoztatta egy kiválasztott komponensről. Összetettebb elemek esetében (például listás, vagy táblázatos megjelenítés) felmerülhetnek további problémák is, amik a fentebb említett módszerrel nem orvosolhatók, ezért az ilyen elemek irányíthatóságára további gesztusok bevezetésére is szükség volt.

Mivel mobiltelefonos környezetben még mindig az egyik legnépszerűbb kommunikációs formának az SMS üzenetküldés tekinthető, ezért egy olyan üzenetkezelő rendszer és a hozzátartozó beszélő billentyűzet kifejlesztése mellett döntöttem, ami a fent említett metódust használja, és rezgés és hang alapú visszacsatolás formájában tájékoztatja a felhasználót a képernyőn történtekről. Az utóbbi alkalmazásnak nagy előnye, hogy nem csak az üzenetkezelő rendszerrel képes együttműködni, de képes operációs rendszer szinten beépülni, így más alkalmazások számára is elérhető.

Az elkészült rendszert próbáltam minél egyénre szabhatóbbá tenni, ezért a felhasználó számára meghagytam annak lehetőségét, hogy a beállítások menüben mind a szövegfelolvasó paramétereit, mind a nyelvi beállításokat, mind a színsémákat, mind pedig a betűméreteket dinamikusan változtathassa.

Az elkészült alkalmazásokat a moduláris és stressz tesztek mellett 5 látó és 1 vak felhasználó segítségével teszteltem. A teszteredmények és a vak felhasználóval történt konzultáció alapján elmondható, hogy a módszer jó kezdeményezés, ami használható megoldást nyújthat érintőképernyős környezetben a látássérült emberek problémáira.

10. Köszönetnyilvánítás

Szeretném megköszönni konzulenseimnek Dr. Németh Gézának és Tóth Bálint Pálnak, hogy a módszer és a demonstrációs alkalmazások kidolgozása során szakmai tapasztalatukkal, és ötleteikkel segítették munkámat. Továbbá szeretném megköszönni a segítséget Ecsedi Csabának is, aki hasznos tanácsokkal és észrevételekkel látott el a tesztelés során.

11. Irodalomjegyzék

[1] Tóth B., Németh G., Hidden Markov Model Based Speech Synthesis System in Hungarian, Infocommunications Journal, Volume LXIII, 2008/7, July 2008, pp.: 30-34

[2] Tausz K, Lakatos M, A fogyatékos emberek helyzet, Statisztikai Szemi, 82. évfolyam, 2004. 4. szám, pp.: 370-391.

[3] Braille, http://en.wikipedia.org/wiki/Braille (2012. október)

[4] Wu, M. and Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In Proc. UIST '03, ACM Press, pp.:193-202.

[5] Gartner, http://www.gartner.com/it/page.jsp?id=2120015 (2012. október)

[6] Kane S. K., Bigham J. P. and Wobbrock J. O.. Slide Rule: Making Mobile Touch Screens Accessible to Blind People Using Multi-Touch Interaction Techniques. ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '08). New York: ACM Press, pp.:73-80.

[7] Android for the blind, http://www.codefactory.es/en/products.asp?id=415 (2012. október)

[8] VoiceOver, http://www.apple.com/accessibility/voiceover/ (2012. október)

[9] Mihajlik P., Fegyó T., Németh B., Tüske Z., and Trón V., „Towards Automatic Transcription of Large Spoken Archives in Agglutinating Languages-Hungarian ASR for the MALACH Project" in Proc. of TSD 2007, under the Springer (Heidelberg) series LNCS-LNAI, „Text, Speech and Dialogue", V. Matousek and P. Mautner (Eds.), Pilsen, Czech Republic, September 2007, pp.: 342-350

[10] Nuance, http://www.nuance.com/index.htm (2012. október)

[11] BlindType, http://www.BlindType.com (2012. október)

[12] M. Silfverberg, I. Scott MacKenzie, P. Korhonen, „Predicting Text Entry Speed on Mobile Phones", CHI 2000 April 1-6 2000, pp.: 9-16

[13] SmartBraille, https://play.google.com/store/apps/details?id=jp.tmhouse.SmartBraille&hl=en

[14] Web Accessibility Initiative (WAI), http://www.w3.org/WAI/ (2012. október)

[15] US research by Acision shows SMS is still the king of messaging, http://www.acision.com/News-and-Events/Press-Releases/All-Destinations/2012/US-research-by-Acision-shows-SMS-is-still-the-king-of-messaging.aspx (2012. október)

[16] Android-alapú szoftverfejlesztés, ISBN: 978-963-9863-27-9

[17] UML 2 Semantics and Applications, Canada, ISBN: 978-0-470-40908-4

[18] Dashboards|Android Developers, http://developer.android.com/about/dashboards/index.html (2012. október)

[19] Avraham Leff, James T. Rayfield, Web-Application Development Using the Model/View/Controller Design Pattern, 2001, pp.: 118-127

[20] Input Method, : http://developer.android.com/guide/topics/text/creating-input-method.html#InputMethodLifecycle (2012. október)

[21] UI/Application Exerciser Monkey, http://developer.android.com/tools/help/monkey.html (2012. október)