Entwicklung mobiler Anwendungen für Blinde

Mandy Goram, TREvisto GmbH

Mobile Endgeräte wie Smartphones und Tablets sind im Laufe der letzten Jahre zu einem nahezu unverzichtbaren und nützlichen Begleiter im Alltag geworden – schnell einen Begriff recherchieren und sich mithilfe einer App oder eines Online-Dienstes in einer unbekannten Stadt orientieren. Die Großzahl der Apps ist jedoch nicht barrierefrei gestaltet und somit nahezu unzugänglich für Blinde und stark Sehbeeinträchtigte. Dieser Artikel richtet sich an Designer sowie Entwickler und erläutert die Accessibility-Funktionalitäten von Android.

Für Blinde und stark Sehbeeinträchtigte sind die visuelle Wahrnehmung und die Aufnahme visueller Informationen nicht gegeben beziehungsweise sehr stark eingeschränkt. Daher können Darstellungen und Inhalte einer Anwendung nicht in der klassischen Art und Weise vermittelt werden. Man benötigt alternative Mechanismen in der Mensch-Computer-Interaktion und muss diese in geeigneter Weise implementieren.

Bei mobilen Endgeräten bieten sich meist die Sprach- und Tonausgabe sowie Vibrationsmechanismen an. Für die Aufbereitung und Wiedergabe von inhaltlichen Informationen stehen in diesem Fall auditives, haptisches und taktiles Feedback zur Verfügung. Bei auditivem Feedback handelt es sich im Wesentlichen um Sprach- und Soundwiedergabe. Unter Haptik versteht man die Berührung und aktive Erkundung von Oberflächen. Dadurch können Formen und oberflächliche Strukturen erkannt werden, etwa die Dimensionen und die Form eines mobilen Endgerätes. Die taktile Wahrnehmung ist unter anderem durch die Empfindung von Temperatur, Festigkeit, Schmerz oder Vibration geprägt, beispielsweise bei einer Auswahlbestätigung durch eine kurze Vibration.

Das haptische Feedback, im Englischen auch „Forced Feedback" genannt, spielt bei der Entwicklung virtueller Realitäten eine bedeutende Rolle und wird zur aktiven Kraftrückkopplung verwendet. Bei Smartphones und Tablets steht hingegen das taktile Feedback im Vordergrund. Daher haben

Version	Accessibility-Funktionalitäten
Donut (1.6), Eclair (2.1), Froyo (2.2), Gingerbread (2.3)	Bereitstellung von Screenreadern („TalkBack", „Spiel", „MobileAccessibility")
Honeycomb (3.0)	Verbesserte Web-Zugänglichkeit
Ice Cream Sandwich (4.0)	Explore by touch (Sprachwiedergabe der Komponenten-Beschreibung) Aktivierung „TalkBack" bei erstem Systemstart möglich Systemweite Schriftgrößen-Anpassung
Jelly Bean (4.1)	Überwachung und Steuerung komplexer Gesten Unterstützung von Braille-Services via „BrailleBack" Zoom von Bildschirm-Ausschnitten Traversieren von Texten Anpassung der Navigation (Fokussieren von Objekten) Ausstattung von Widgets mit logischen Modellen
Jelly Bean (4.3)	Verarbeitung von Tastatureingaben „Copy & Paste"-Funktion Verbesserte Web-Zugänglichkeit

Tabelle 1: Accessibility-Funktionalitäten der Android-Versionen

entsprechende Mechanismen ihren Weg in das Android-API gefunden.

Accessibility in Android

Das Android-API bietet mittlerweile umfangreiche, weit entwickelte Schnittstellen und Konzepte zur Entwicklung barrierefreier Anwendungen an. Da diese Funktionalitäten erst nach und nach über das API bereitgestellt wurden, war die barrierefreie Entwicklung in den Anfangsjahren sehr aufwändig und aufgrund der unterschiedlichen Geräte sehr unzuverlässig. Während bis zur Version 2.3. („Gingerbread") lediglich die Screenreader „TalkBack", „Spiel" und „MobileAccessibility" zur Verfügung standen und mit der Version 3.1. („Honeycomb") nur die Web-Zugänglichkeit verbessert wurde, gibt es seit 4.0 („Ice Cream Sandwich") wesentliche Verbesserungen und Neuerungen im Accessibility-API. Es steht beispielsweise der „Explore by Touch"-Modus zur Verfügung, der View-Komponenten gezielt selektieren kann und eine sprachliche Beschreibung wiedergibt. Die Funktionen stehen in Verbindung mit dem mittlerweile zentralen Screenreader „TalkBack". Dazu später mehr.

Mit der Version 4.1 („Jelly Bean") kamen durch das Accessibility-API weitere sehr nützliche Funktionen, die eine einfache, barrierefreie Entwicklung ermöglichen. So stehen Funktionen zur Gesten-Überwachung und -Steuerung bereit, um auf einfachem Wege neue Gesten zu implementieren. Zudem wird erstmals die Verwendung von Braille-Tastaturen durch das Betriebssystem direkt ermöglicht. Dazu sind Braille-Services via „BrailleBack" in den „TalkBack"-Modus integriert. Dies ermöglicht die Ein- und Ausgabe von Zeichen ohne aufwändige Implementierung und ohne Transformation der Informationen. *Tabelle 1* gibt einen Überblick zu den wichtigsten bereitgestellten Funktionen und Verbesserungen in Android.

Wie eine zusammengefasste Statistik vom August 2014 zur Verbreitung der Android-Versionen zeigt, laufen mehr als 75 Prozent aller Android-Geräte mit einer Version 4.0 oder höher *(siehe Abbildung 1)*. Somit können die meisten Geräte mit barrierefreien Apps arbeiten, die die Standard-Funktionalitäten des API verwenden.

Bedienhilfen „Out of the box"

Es ist nicht viel Aufwand erforderlich, um eine App für die Screenreader oder die Sound- beziehungsweise Vibrationswiedergabe zugänglich zu machen. Werden zur Erstellung der App Standard-Komponenten verwendet, können die von Android bereitgestellten Mechanismen einfach darauf zugreifen. „TalkBack", „SoundBack", „KickBack" und „BrailleBack" sind robuste Schnittstellen zwischen dem Gerät und dem Anwender, wodurch dem Entwickler eine aufwändige Eigenentwicklung erspart bleibt. Bei der Verwendung eigener Nicht-Standard-Komponenten in der View muss aber gegebenenfalls ein eigener Feedback-Mechanismus entwickelt werden. Konkret kann es sich dabei um spezielle Gesten oder Sprachausgaben handeln.

Die zentrale Accessibility-Komponente ist „TalkBack". Die Funktionen sind über das Menü „Bedienhilfen" aktivierbar. Ab der Version 4.0 kann bereits beim ersten Start eines Geräts dessen Aktivierung erfolgen; dadurch verändern sich die Steuerung und der Auswahlprozess des Geräts grundsätzlich. Der Modus sowie dessen spezielle Erweiterungen ermög-

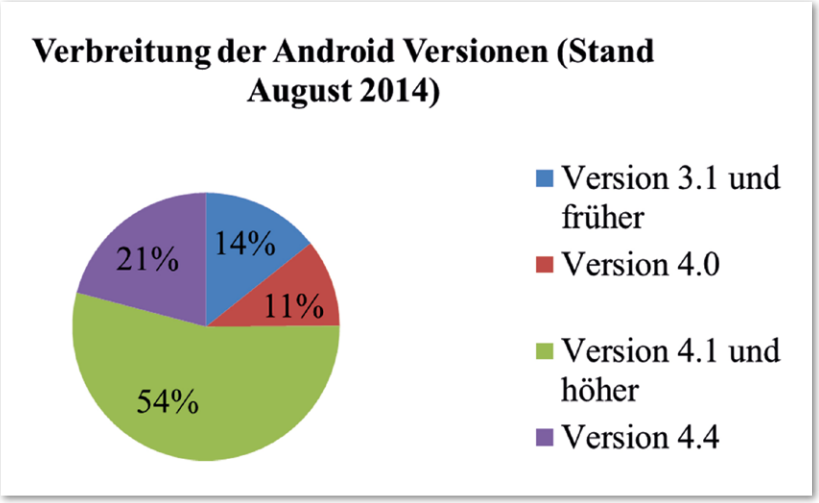

Abbildung 1: Verbreitung der Android-Versionen

```
<Button
  android:id="@+id/play_button"
  android:contentDescription="@string/play"/>
```

Listing 1: „contentDescription" eines View-Elements

```
<EditText
  android:id="@+id/search_criteria"
  android:hint="@string/search_hint"
  android:inputType="text" />
```

Listing 2: „android:hint" eines Textfelds

```
<Button android:id="@+id/focus_button"
  android:focusable="true"
  android:nextFocusUp="@id/edit"
  ... />
```

Listing 3: Setzen von „focusable" im XML-File

```
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
…
textToSpeech = new TextToSpeech(context, this);
  textToSpeech.setLanguage(Locale.GERMANY);
textToSpeech.speak("Hallo!", TextToSpeech.QUEUE_FLUSH, null);
…
```

Listing 4: Wiedergabe mittels TTS

lichen Blinden eine selbstständige Bedienung des Android-Geräts über permanentes Feedback des Betriebssystems an den Anwender. Dadurch können verschiedene Einstellungen vorgenommen und einzelne Apps ausgewählt werden. Wie sich eine ausgewählte App nach dem Start verhält, hängt aber im Wesentlichen von deren Implementierung ab. Wurden barrierefreie Elemente nur unzureichend oder gar nicht berücksichtigt, ist die Bedienung durch einen Blinden nicht möglich und die inhaltliche Erschließung sehr schwer bis unmöglich.

Zugänglichkeit von Oberflächenelementen

TalkBack verwendet die „contentDescription" eines View-Elements, um Informationen zum Inhalt und zum Aufbau einer View wiederzugeben *(siehe Listing 1)*. Ist diese für ein Element gesetzt, kann der Screenreader sie auslesen. Das Setzen kann sowohl für Standard- als auch für eigene Komponenten erfolgen. Bei rein optischen View-Elementen sollte diese allerdings nicht gesetzt werden. Dadurch wird verhindert, dass inhaltlich irrelevante Elemente angesprochen werden. Dies könnte bei einem Blinden zu Irritationen und Verwirrung führen, da er selbst die Bedeutung nicht ohne entsprechende Rückmeldungen erfassen kann.

Bei EditBoxen muss für eine entsprechende Funktionalität anstelle der „contentDescription" ein „android:hint" verwendet werden. Dadurch erfolgt bei Auswahl eines Eingabefelds die Wiedergabe von Feldbezeichnung und -inhalt *(siehe Listing 2)*.

Für den „Explore by Touch"-Modus ist es wichtig, dass die View zugängliche Komponenten besitzt. Hierfür muss jede anwählbare Komponente „focusable = true" deklariert haben. Dies kann im XML-File gesetzt sein *(siehe Listing 3)* oder nachträglich im Code geändert werden. Dazu können die Methoden „setFocus", „isFocusable" und „requestFocus" zum Einsatz kommen.

Zur Anwahl eines Elements durch den Anwender verfügt der „Explore by Touch"-Modus über eine spezielle Gesten-Steuerung. Mit einer einfachen Wischgeste wird das nächste fokussierbare Objekt in der View ausgewählt. Die Werte „nextFocusRight", „nextFocusDown", „nextFocusLeft" und „nextFocusUp" verfeinern die Ansteuerung einzelner Elemente.

Sprachausgabe und Sprachsteuerung

Man ist jedoch nicht ausschließlich auf den Screenreader angewiesen, um Komponenten und Inhalte vorlesen zu lassen. Die Text-to-Speech-Schnittstelle (TTS) integriert individuelle Sprachausgaben in die Anwendung. Dazu ist ein Objekt mit einem vorzulesenden Text zu erzeugen und wiederzugeben; die Verwendung von „TalkBack" entfällt. Je nach Zweck kann die Wiedergabe von Text durch eine aktive Interaktion des Anwenders mit der Anwendung erfolgen oder zeitgesteuert ablaufen. Diese Feinheiten sind ganz dem Entwickler überlassen *(siehe Listing 4)*.

Es geht aber auch umgekehrt mit Speech-to-Text (STT). Der SpeechRecognizer wandelt Sprache in Text um. Im Zuge dessen lassen sich unter anderem Sprachkommandos integrieren, die eine Sprachsteuerung der Anwendung erlauben. Der SpeechRecognizer hat jedoch einen entscheidenden Nachteil, er benötigt eine Internet-Verbindung zum Server. Das API stellt damit lediglich eine Service-Schnittstelle zur Verfügung. Ist das mobile Endgerät offline, kann die Funktion nicht verwendet werden.

Eine Alternative dazu ist das Projekt „pocketsphinx". Es wurde für leichtgewichtige mobile Anwendungen entwickelt und ist eines der wenigen Projekte zur Spracherkennung und -steuerung, das auch offline verwendet werden kann. Die Programm-Bibliothek lässt sich in ein Android-Projekt integrieren und ohne Internet-Anbindung ausführen. Die Integration in die App gestaltet sich zunächst etwas aufwändig. Erst mal muss man die Bibliothek überhaupt zum Laufen bekommen, dann sind Grammatiken und das Wörterbuch selbst zu erstellen. Ein Tool unterstützt beim Anlegen des Wortschatzes, indem es die Informationen für „pocketsphinx" lesbar in eine Datei übersetzt.

Allgemeines Oberflächen-Design

Neben der eigentlichen Implementierung und Ausgestaltung des Codes spielt bei der Entwicklung barrierefreier Anwendungen auch das Design der Oberfläche eine bedeutende Rolle. Dabei gilt es, die Verwendung von Freitext-Feldern weitestgehend zu vermeiden, sofern diese nicht zwingend erforderlich sind. Bei dis-

kreten Werten sollten vor allem Checkboxen oder Radiobuttons zum Einsatz kommen. Das unterstützt auch einen normalen Anwender bei der Bedienung der App und vermeidet fehlerhafte Eingaben und Systemzustände.

Generell sollte die Anwendung nicht mit Informationen überladen werden. Zudem ist eine einfache und übersichtliche Navigationsstruktur sehr wichtig. Jede View sollte genau eine Aufgabe erfüllen, etwa die Eingabe eines Suchkriteriums oder das Anlegen eines Kontakts. Um auch Sehbeeinträchtigte bei der Bedienung zu unterstützen, sollten Schriftgrößen individuell vergrößerbar sein und wichtige Informationen zumindest eine angemessene Textgröße besitzen. Darüber hinaus ist die Verwendung von Kontrasten sehr wichtig, um auch Farbfehlsichtigen die inhaltliche Zuordnung einfach zu ermöglichen. Diese Gestaltungsgrundsätze beruhen auf den allgemeinen Forderungen zur ergonomischen und benutzerfreundlichen Gestaltung von Anwendungssoftware.

Fazit

Das Android-API bietet mittlerweile sehr umfangreiche Möglichkeiten zur barrierefreien Entwicklung von Apps. Nur wenige beziehungsweise gar keine Funktionen müssen dafür eigens entwickelt werden. Es ist also nicht notwendig, eine App speziell nur für Blinde zu implementieren.

Bestehende Anwendungen können durch kleine Anpassungen der View-Elemente barrierefrei gestaltet werden, ohne die bisherigen Funktionen einzuschränken. Lediglich bei der Abwärtskompatibilität zu älteren Android-Versionen sind gegebenenfalls Mechanismen eigenständig zu integrieren. Wie die aktuelle Statistik zeigt, kann jedoch der Großteil der Endgeräte die meisten Accessibility-Funktionalitäten verwenden.

Es gibt also keinen Grund, die Barrierefreiheit bei der Entwicklung von Android-Apps außer Acht zu lassen. Das API kommt dem Entwickler sehr weit entgegen und bietet auch dem Anwender für den normalen Einsatz der Anwendung einen echten Mehrwert, wie die Sprachausgabe, die Vorlesefunktionen und die Sprachsteuerung. Ein einfaches und intuitives Oberflächen-Design trägt zudem zur Benutzerfreundlichkeit und Bedienbarkeit bei, was letztendlich allen Anwendern zugutekommt.

Weiterführende Links

1. http://developer.android.com/guide/topics/ui/accessibility/apps.html
2. https://developer.android.com/tools/testing/testing_accessibility.html
3. https://support.google.com/accessibility/android/#topic=6007234
4. http://cmusphinx.sourceforge.net/wiki/tutorialandroid

Mandy Goram
mandy.goram@trevisto.de

Mandy Goram ist eine erfahrene Front- und Back-End-Entwicklerin. Sie arbeitet als IT-Beraterin für die TREvisto GmbH in Nürnberg. Ihre aktuellen Schwerpunkte sind Daten-Analyse und Stammdaten-Management (MDM). Privat beschäftigt sie sich unter anderem mit der Entwicklung von Android Apps und der Thematik "Usability".

http://ja.ijug.eu/15/1/19

Enterprise Apex: Die Neuerungen von Apex 5.0

Application Express (Apex) ist als Teil der Oracle-Datenbank seit mehr als zehn Jahren verfügbar und wird in etlichen IT-Landschaften in großem Stil eingesetzt. Einige Neuerungen von Apex 5.0 eignen sich besonders für „Enterprise Apex".

Der Page Designer ist mit Abstand die auffälligste Änderung in der neuen Version und betrifft ausnahmslos alle Apex-Entwickler. Über diese große Neuerung hinaus bringt Apex 5.0 noch eine Menge neuer Funktionen für den Unternehmenseinsatz.

Das Universal Theme und die Theme-Styles erlauben es, einer Apex-Anwendung unterschiedliche „Look & Feels" zuzuweisen, ohne das Theme austauschen zu müssen. Der HTML-Code der Templates wird nicht geändert. Vielmehr wird dem Theme ein Style zugewiesen, der im Wesentlichen aus CSS-Angaben besteht. Mit diesen CSS-Angaben lassen sich Farbe, Schriftart, Linienbreite und mehr ändern. Die Templates bleiben unberührt.

Bisher waren die statischen Dateien eines Apex-Workspaces in Bilder, CSS und sonstige statische Dateien unterteilt. Nun sind sie zusammengefasst. Entwickler können zudem zu jeder Datei einen Verzeichnispfad hinterlegen: Das ist sehr nützlich, wenn mehrere Dateien hochgeladen werden, die sich gegenseitig mit relativen URL-Pfaden referenzieren. Auch können statische Dateien als „.zip"-Dateien hochgeladen und innerhalb von Apex automatisch entpackt werden. Dafür nutzt das Entwickler-Tool das PL/SQL-Paket „Apex_ZIP", das auch das Ein- und Auspacken von ZIP-Archiven in der Datenbank ermöglicht.

Ein weiteres PL/SQL-Paket dürfte nicht nur für Apex-Entwickler hochinteressant sein: „Apex_JSON" gibt dem Anwender eine PL/SQL Schnittstelle zur Arbeit mit JSON-Daten. Diese kann auf 12c-Datenbanken mit der neuen JSON-Funktionalität auf SQL-Ebene kombiniert werden.

Einen umfassenden Einblick in die neuen Funktionen von Apex 5.0 gibt die vom Apex-Entwicklerteam gepflegte Liste der neuen Funktionen *(siehe „https://apex.oracle.com/pls/apex/f?p=65339")* sowie das Statement of Direction *(siehe „http://www.oracle.com/technetwork/developer-tools/apex/application-express/apex-sod-087560.html")*.