Leseprobe

Der Weg zur eigenen Android-App beginnt hier, garniert mit jeder Menge Lesespaß! Ihr erstes App-Projekt ist dabei kein Kinkerlitzchen: Sie programmieren gemeinsam mit Uwe Post ein komplettes Spiel. Diese Leseprobe hilft Ihnen bereits beim Einstieg und stellt Ihnen die Android-Welt und Android Studio 3 vor. Außerdem lernen Sie, wie Sie Programmierfehlern schnell auf die Schliche kommen.

Uwe Post

Android-Apps entwickeln für Einsteiger

388 Seiten, broschiert, 7. Auflage, Januar 2018
24,90 Euro, ISBN 978-3-8362-6139-5

www.rheinwerk-verlag.de/4586

Kapitel 1
Einleitung

»Jede hinreichend fortgeschrittene Technologie ist von Magie nicht mehr zu unterscheiden.«
(Arthur C. Clarke)

Seit Ende 2008 existiert mit Android ein Betriebssystem für Smartphones und andere handliche Geräte, das sich schnell Millionen Freunde gemacht hat. Der Hauptgrund dafür sind die unzähligen Apps, die sich in Sekundenschnelle installieren lassen. Obwohl diese kleinen Programme auf den ersten Blick unscheinbar wirken, haben sie das Leben vieler Menschen verändert – und bei vielen den Wunsch geweckt: Das will ich auch!

Sie gehören zu diesen Menschen, können aber noch nicht programmieren? Gratulation! Dann haben Sie sich für das richtige Buch entschieden.

1.1 Für wen ist dieses Buch?

Heerscharen von Programmierern haben sich auf die Android-Entwicklung gestürzt. Im Handumdrehen bastelten sie erste Apps zusammen, denn die Einstiegshürde ist niedrig. Folglich kann es auch für Noch-Nicht-Programmierer nicht allzu schwer sein, mit der App-Entwicklung zu beginnen. Es ist wirklich nicht besonders kompliziert, und die nötige Programmiersprache – Java – lernen Sie praktisch nebenbei.

Selbst wenn Sie noch nie programmiert haben, können Sie sich am Ende dieses Buches zur wachsenden Gemeinde der Android-Entwickler zählen.

Wer bereits programmieren kann, wird in diesem Buch ein paar Kapitel überschlagen können, im Anschluss daran aber anspruchsvolle Android-Techniken kennenlernen – begleitet von einigen heißen Tipps.

Experten werden sich vermutlich hin und wieder über einige Vereinfachungen wundern, die ich im Interesse der Einsteiger vor allem bei der Erklärung von Java vorgenommen habe – drücken Sie einfach ein Auge zu, und konzentrieren Sie sich auf die fortgeschrittenen Technologien und Tipps zur Android-Entwicklung. Dann werden auch Sie dieses Buch nicht als nutzlos empfinden.

1.1.1 Magie?

»Moin auch, die Temperatur beträgt heute 22 °C und die Regenwahrscheinlichkeit 5 %.« Eine Frauenstimme neben dem Kopfkissen weckt Sie mit freundlichen Worten, die Sie selbst ausgesucht haben. Falls Sie wieder einschlafen, versucht die Stimme es kurz darauf etwas lauter und weniger diskret: »Raus aus den Federn, aber zackig!«

Die neue Müslipackung erwähnt Zucker schon an zweiter Stelle – Sekunden später finden Sie heraus, dass der Hersteller die Rezeptur geändert hat. Ihr Missfallen, per Touchscreen im sozialen Netzwerk veröffentlicht, hat schon das halbe Land durchquert, noch bevor Sie das Fahrrad aus der Garage geholt haben. Eile ist nicht geboten, denn Sie wissen bereits, dass die S-Bahn spät dran ist. Im Zug nutzen Sie die Zeit, um Ihren breitschultrigen Fantasyhelden im Umgang mit seinem neuen Kriegshammer zu trainieren.

Beim Einkaufen verkündet Ihre Armbanduhr, dass es Ihre Lieblingsschokolade im neuen Supermarkt ein paar Straßen weiter zum Einführungspreis gibt. Glücklicherweise ist der Laden auf Ihrem elektronischen Stadtplan eingezeichnet – nicht aber der Hundehaufen, in den Sie treten, weil Sie nur auf das Handydisplay starren.

Jetzt hätten Sie gerne Ihren Kriegshammer – und an diesem Punkt stoßen Sie dann doch an die Grenze zwischen Technologie und Magie.

Überflüssig zu erwähnen, dass die Generationen Ihrer Eltern und Großeltern den größten Teil dieser Geschichte als Ausgeburt Ihrer überbordenden Fantasie bezeichnen würden. Sie wissen es besser, denn Sie halten ein Stück Technologie in den Händen, dessen Fähigkeiten mehr als nur erstaunlich sind. Besser noch: Sie beherrschen sie. Sie erweitern sie. Sie fügen weitere Magie hinzu – solange der Akku reicht.

1.1.2 Große Zahlen

Zum Zeitpunkt der Drucklegung dieser Auflage (Ende 2017) wird weltweit schätzungsweise eine Million neuer Android-Geräte aktiviert – *jeden Tag*. Milliarden von Android-Geräten wurden auf der ganzen Welt insgesamt bereits in Betrieb genommen (aber natürlich werden auch ein paar ausrangiert). Seit vier Jahren stagniert die Zahl, da eine Sättigung erreicht ist. Geräte mit dem pummeligen grünen Roboter als Maskottchen sind längst Alltagsgegenstände geworden.

Über drei Millionen Apps sind in *Google Play* verfügbar, die meisten davon können Sie kostenlos auf Ihr Smartphone herunterladen und benutzen (siehe Abbildung 1.1). Das funktioniert einfach und schnell. Freilich ist ein großer Teil der erhältlichen Apps nutzlos oder von unterirdischer Qualität.

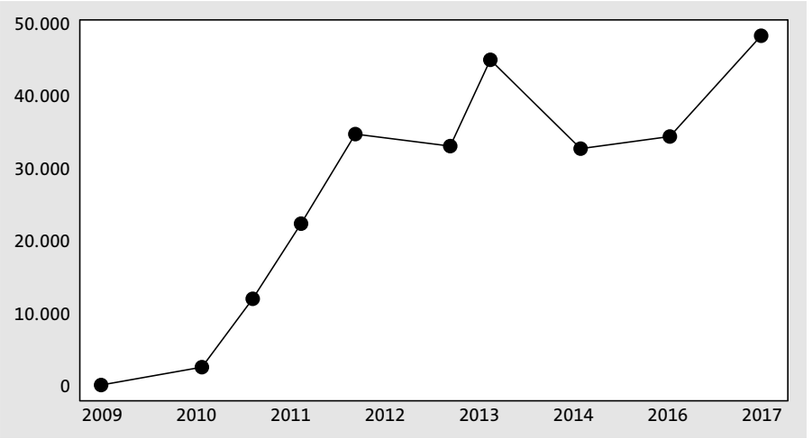

Abbildung 1.1 Jeden Monat erscheint eine fünfstellige Anzahl neuer Apps in Google Play.

Sehr hoch ist daher noch eine andere Zahl: die Anzahl der wieder deinstallierten Apps. Anschauen kostet nichts außer ein paar Minuten Zeit (Flatrate vorausgesetzt), Löschen noch weniger. Allein schon aufgrund des begrenzten Speichers üblicher Android-Geräte müssen Sie davon ausgehen, dass das Entsorgen einer App häufig zu ihrem Lebenszyklus gehört.

Wenn Sie jenen unzähligen Apps eigene hinzufügen möchten, die nicht so schnell wieder deinstalliert werden, ist dieses Buch für Sie genau richtig.

1.1.3 Technologie für alle

Auch die Handys des letzten Jahrhunderts waren prinzipiell programmierbar. Sie verfügten nicht über Bewegungssensoren oder Satellitenempfänger – grundsätzlich aber enthielten sie eine ähnliche Digitalelektronik wie Android-Handys: Mikroprozessoren, Speicher und Schnittstellen, also Hardware, auf der Software lief und erstaunliche Dinge bewirkte. Es gibt jedoch einen entscheidenden Unterschied: Nur ausgewiesene Experten, die über teure Spezialgeräte verfügten, konnten solche Handys programmieren (siehe Abbildung 1.2).

Bei Android-Smartphones sieht die Sache anders aus. Der Hersteller Google hat sich eine einfache Strategie überlegt: Je mehr Apps es gibt, desto interessanter sind Smartphones, und desto mehr Menschen kaufen welche. Also brauchen wir möglichst viele Menschen, die Apps entwickeln. Daher gestalten wir das Entwickeln von Apps so einfach, dass es jeder kann!

Abbildung 1.2 Nicht jedes Handy vor dem Smartphone-Zeitalter war nur zum Telefonieren geeignet – bloß konnten weder Sie noch ich Apps dafür entwickeln.

PC und Laptop sind völlig ausreichende Arbeitsgeräte, um mit der App-Entwicklung zu beginnen. Die nötige Software ist kostenlos. Als Programmiersprache kommt Java zum Einsatz – eine Sprache, die leicht zu lernen ist und die nicht jeden kleinen Fehler mit rätselhaften Abstürzen oder Datenverlust bestraft. Java ist so verbreitet, dass es jede Menge Bücher darüber gibt – und Bekannte, die man fragen kann, wenn man nicht weiterkommt.

Wenn Sie ein solcher Bekannter werden wollen, dann ist dieses Buch für Sie ebenfalls genau richtig.

1.1.4 Die Grenzen der Physik

Einstein hat unter anderem herausgefunden, dass Masse den Raum krümmt, dass Raum und Zeit zusammenhängen und dass Genies auch ohne Friseure auskommen. Mithilfe eingebauter Sensoren kann ein Smartphone zwar das Schwerefeld der Erde messen, um oben und unten auseinanderzuhalten, merklich beeinflussen kann aber selbst das massivste Gerät weder Raum noch Zeit (siehe Abbildung 1.3), sodass Zeitmaschinen weiterhin nur in der Fantasie einiger Science-Fiction-Autoren existieren.

Daher kommen Sie auch mit dem modernsten Smartphone nicht umhin, eine Menge Zeit zu investieren, bevor Sie Ihre ersten wirklich magischen Apps fertigstellen werden. Zu ersten vorzeigbaren Apps werden Sie schnell kommen, jedoch: Die Android-Technologie ist sehr mächtig, sonst könnte sie kaum ihre Wunder bewirken. Manchmal werden Sie vor einem Rätsel stehen, aber ich werde Ihnen die nötigen Schlüssel in die Hand drücken, um all jene verborgenen Schatztruhen zu öffnen, die am Wegesrand stehen, und Sie bekommen auch genug Plastiktüten, um all die Tretminen zu entfernen, die rücksichtslos in der Gegend deponiert wurden.

Abbildung 1.3 Selbst besonders schwere Smartphones wie dieses Motorola Milestone krümmen nicht messbar den Raum (beachten Sie die untere Skala der Waage).

Auch wenn es eine lange, streckenweise beschwerliche Reise wird: Die gute Nachricht ist, dass die Steigung des Weges nicht mit jener der Zugspitzbahn (bis zu 250 Promille) zu vergleichen ist. Die Android-Entwicklung hat eine sehr flache Lernkurve, denn die wirklich kniffligen Dinge nimmt uns das System mit dem freundlichen, etwas pummeligen Roboter ab. Sie werden nur wenige Minuten benötigen, um Ihre erste App zu schreiben.

Und wenn Sie darüber staunen möchten, wie einfach die Android-Entwicklung ist – auch dann ist dieses Buch für Sie genau richtig.

1.2 Unendliche Möglichkeiten

Smartphones bieten Unmengen an Funktionen – genau das hat ihnen schließlich ihren Namen eingebracht. Sie sind *smart*. Apps können die bereitgestellten Funktionen ausnutzen, ganz nach Bedarf. Natürlich bringt Ihnen dieses Buch nicht jede dieser Funktionen im Detail bei. Aber Sie erhalten die nötige Grundausbildung, um mit ein bisschen Recherche prinzipiell jede Funktion Ihres Smartphones nutzen zu können.

1.2.1 Baukasten

Vielleicht haben Sie als Kind mit einem Metallbaukasten gespielt oder mit Lego. Entwickler gehen nicht anders vor, um einfache Apps zu erstellen: Sie nehmen vorhandene Teile und stecken sie wie benötigt zusammen. Wenn etwas klemmt, liegt ein Hammer bereit.

Wenn Sie sich die Apps auf Ihrem Smartphone in Ruhe anschauen, werden Sie sehen, dass sie eine ganze Reihe von Komponenten gemeinsam haben. Buttons in Apps mögen unterschiedlich groß sein, unterschiedliche Beschriftungen tragen und natürlich verschiedene Wirkungen entfalten, wenn jemand sie drückt. Aber alle sind *Buttons*. Sie müssen sich nicht darum kümmern, wie der Button auf den Bildschirm kommt, wie der Touchscreen funktioniert oder dass der Button unterschiedlich aussehen kann – abhängig davon, ob er deaktiviert oder gerade gedrückt wurde.

Dasselbe gilt für Textbausteine, Eingabefelder, Bilder, Fortschrittsbalken und so weiter. Mit wenigen Mausklicks lassen sich einfache Formulare, Listen und Bildschirmdarstellungen zusammenbauen und justieren. Um solche Benutzeroberflächen mit Leben zu füllen, kommt Java-Programmcode zum Einsatz. Da sich das Android-System um den größten Teil des Ablaufs ganz allein kümmert, sind oft nur wenige Zeilen einfacher Programmanweisungen notwendig, um eine sinnvolle App zu erhalten. So entstehen elektronische Einkaufszettel, Vokabeltrainer oder Zähler für erschlagene Mücken in einem feuchten Sommerurlaub in Skandinavien (wenn Sie glauben, so etwas sei überflüssig, waren Sie noch nie in Schweden).

Viele auf den ersten Blick einfache Apps nutzen Dienste im Internet. Prinzipiell kann fast jede Funktion, die auf einer Webseite zur Verfügung steht, in eine Android-App verpackt werden. Zwar bietet Android auch einen Browser, aber oft ist es im Mobilfunknetz viel effizienter, keine ganzen Webseiten abzurufen, sondern nur die gewünschte Funktion. Auf diese Weise lässt sich beispielsweise sehr einfach ein Synonymwörterbuch (Thesaurus) bauen, das genau so funktioniert wie die entsprechende Funktion auf der Webseite *http://openthesaurus.de* (siehe Abbildung 1.4).

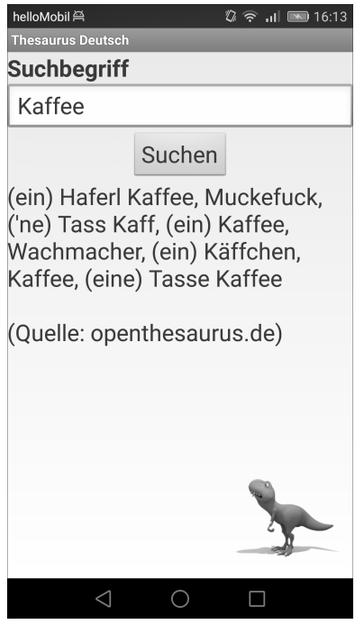

Abbildung 1.4 Frei zugängliche Internetdienste lassen sich leicht in Android-Apps verpacken und ersparen dem Benutzer den Umweg über Browser und Webseite.

Solche öffentlich und kostenlos zugänglichen Internetkomponenten können Sie ähnlich wie die oben aufgeführten visuellen Komponenten als »Blackbox« in Ihre App integrieren, ohne über die internen Vorgänge Bescheid zu wissen oder sie gar selbst bauen zu müssen (freilich müssen Sie unter Umständen eine Erlaubnis einholen).

Softwareentwicklung im 21. Jahrhundert hantiert nicht mehr mit einzelnen Bits und Bytes. Das wäre viel zu aufwendig; Sie würden erst nach Jahren mit einer einfachen App fertig werden. Vielmehr besteht die Herausforderung darin, die richtigen Komponenten zu kennen, die optimal zueinanderpassen, und sie dann so zusammenzusetzen, dass sie fehlerfrei funktionieren.

1.2.2 Spiel ohne Grenzen

Im Laufe dieses Buches werden Sie die Android-Entwicklung zunächst anhand eines Spielprojekts kennenlernen. Das macht mehr Spaß als eine vergleichsweise trockene Synonyme-Anwendung, die meist weder von den Lagesensoren noch vom Kompass be-

sonders profitiert. Außerdem ist der Wow-Effekt größer, und die Themen sind herausfordernder.

Zu den herausragenden Eigenschaften von Smartphones gehören die eingebauten Sensoren. Vielleicht haben Sie gelegentlich in der TV-Serie »Star Trek. Das nächste Jahrhundert« das höchst erstaunliche Gerät namens Tricorder bewundert: eine technische Meisterleistung im handlichen Hosentaschenformat und gleichzeitig ein Deus ex Machina für die Drehbuchautoren, wenn ihnen für eine herbeigeführte Situation keine andere Lösung einfiel. Es gab einmal eine Tricorder-App für Android, die fast das Gleiche konnte, ausgenommen die Explosion beim Überhitzen (wobei ich das offen gesagt nie ausprobiert habe). Allerdings musste der Autor sie nach einer Beschwerde der Rechteinhaber entfernen. Zum spielerischen Kennenlernen der Sensoren eignet sich stattdessen die App *Sensor Box for Android* (siehe Abbildung 1.5), die Sie in *Google Play* leicht unter diesem Namen finden.

Abbildung 1.5 Die »Android Sensor Box« enthält Kugeln, berührungsempfindliche Pflanzen und eine Wasserwaage zum einfachen Kennenlernen der Fähigkeiten Ihres Smartphones.

Übrigens: Wenn Sie Ihr Handy in Ihrem Google-Account aktiviert haben und in einem PC-Browser unter der Adresse *http://play.google.com* nach Apps suchen, können Sie diese per Mausklick auf Ihr Handy schicken.

Eine App wie *Android Sensor Box* ist die einfachste Möglichkeit, die eingebauten Sensoren in Aktion zu erleben. Wählen Sie im Hauptmenü einen Modus aus, z. B. ACCELEROMETER SENSOR. Sie sehen eine auf einem flachen Laminatboden rollende Kugel, die der Schwerkraft folgt. Tippen Sie auf die Knöpfe DATA oder GRAPH, um sich die numerischen Ursachen für die Bewegung der Kugel anzusehen. Diese Zahlen werden vom Beschleunigungssensor geliefert. Wenn Sie das Handy flach auf den Tisch legen, sehen Sie bei Z einen Wert von etwa 9 – das entspricht der Erdbeschleunigung von 9,81 m/s^2 Richtung Fußboden (oder Richtung Erdmittelpunkt, um genau zu sein). Halten Sie das Gerät aufrecht, wird Z den Wert 0 annehmen, Y dafür den Wert 9.

Die Werte sind zwar nicht genau genug für ernsthafte physikalische Messungen, aber sie genügen, um beispielsweise ein Spiel zu simulieren, in dem Sie eine Kugel durch Neigen des Geräts durch ein Labyrinth manövrieren. Auch Spiele, die das Handy als Steuerrad eines Autos oder anderen Gefährts verwenden, nutzen diese Sensoren.

Außerdem können Sie mit den Beschleunigungssensoren Erdbeben erkennen: Legen Sie das Handy flach hin, schalten Sie in den Modus GRAPH, und schlagen Sie mit der Faust auf den Tisch (bitte nicht zu fest, meine Versicherung zahlt ungern für dabei auftretende Schäden). Sie sehen einen deutlichen Ausschlag in der Fieberkurve.

Probieren Sie als Nächstes den Magnetfeldsensor aus: Wählen Sie das Icon mit dem Magnet, und halten Sie das Handy dann über ein Objekt aus Metall, beispielsweise eine Schachtel Schrauben oder Besteck. Besonders spannend sind 1-Euro- und 2-Euro-Münzen. Sie werden sehen, dass Münzen sehr unterschiedlich magnetisiert sind, je nach ihrem bisherigen Schicksal (siehe Abbildung 1.6). Mit etwas Glück können Sie auf diese Weise einen Schatz finden, allerdings nur, wenn dieser höchstens 1 cm tief vergraben ist – viel empfindlicher ist der Sensor nicht.

Ohne Metall in der Nähe funktioniert der Magnetfeldsensor wie ein Kompass, was sich zahllose Apps in *Google Play* zunutze machen. Darauf werden wir in einem späteren Kapitel noch zurückkommen.

Der akustische Sensor (auch bekannt als Mikrofon) ist eine hübsche Spielerei, aber vom eingebauten Mini-Mikro kann man natürlich keine hohe Auflösung erwarten. Zudem ist Android für seine fürchterlich langsame Audioverarbeitung berüchtigt, was viele Anwendungen sinnlos macht. Sollten Sie auf die Idee kommen, mit dem Smartphone die Lautstärke eines startenden Flugzeugs in Dezibel zu messen, nehmen Sie bitte einen Gehörschutz mit zum Flughafen.

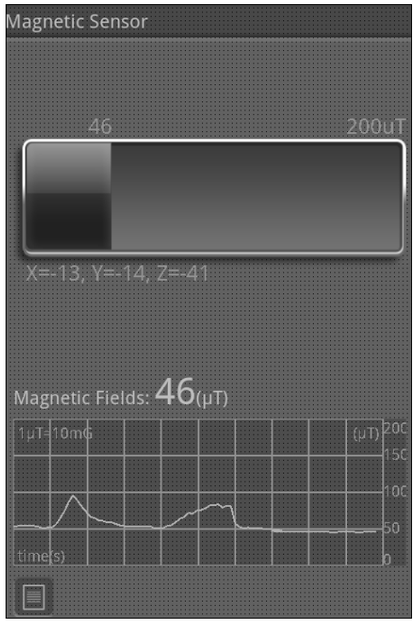

Abbildung 1.6 Mit dem Scannen von unterschiedlichen Münzen kann man sich stundenlang beschäftigen. Ich habe hier drei verschiedene Euro-Stücke der Reihe nach gescannt. Das magnetisch schwache ganz rechts war ein spanisches, aber ich bezweifle, dass man vom Magnetfeld auf die Herkunft schließen kann.

Für viele Anwendungen sehr spannend ist die Positionsmessung. Schalten Sie GPS (Global Positioning System) im Handy an, um auf wenige Meter genaue Geokoordinaten zu erhalten.

Die *Android Sensor Box* ist nicht dazu geeignet, sich mit den globalen Satelliten zu unterhalten, verwenden Sie stattdessen einfach *Google Maps*. Das zeigt Ihnen zwar nicht ohne Weiteres die genauen Geokoordinaten, aber darauf kommt es nicht an. Wir werden später auf diese Details zurückkommen.

Leichten Verfolgungswahn kann man empfinden, wenn man sogar ohne eingeschaltete Satellitenmessung eine relativ genaue Ortsangabe erhält. Es erstaunt auf den ersten Blick, dass das Handy anhand der Mobilfunkzelle und in der Nähe befindlicher WLAN-Router ziemlich genau weiß, wo es sich befindet. Einerseits ist das überaus praktisch, weil im Gegensatz zu GPS praktisch keine Energie verbraucht wird, andererseits freuen sich die Geheimdienste bestimmt außerordentlich über diese hilfreichen Daten.

Derzeit ist die beliebteste Anwendung der Positionsbestimmung mit Smartphones vermutlich das Geocaching (*www.geocaching.com*), eine ziemlich harmlose Schnitzeljagd, die neuerdings sogar Kinder zur freiwilligen Teilnahme an einem Spaziergang animieren kann.

1.2.3 Alles geht

Wenn Sie sich die Funktionen von Smartphones vor Augen halten, werden Sie schnell einsehen, dass es eigentlich nur eine Grenze für mögliche Anwendungsfälle gibt: Ihre Fantasie.

Smartphones sind eine wenige Jahre alte Technologie, die drauf und dran ist, bisher übliche Mobiltelefone in der Hosentasche der meisten Menschen komplett abzulösen. Selbst Science-Fiction-Autoren oder Zukunftsforscher grübeln manchmal so lange über die neuen Möglichkeiten nach, dass ein inspirierter Entwickler an seinem Schreibtisch viel schneller damit fertig ist, eine revolutionäre App zu bauen. Indem Sie dieses Buch lesen, sind Sie mittendrin in dieser Entwicklung – und können sie künftig mitgestalten. Was für ein Spaß!

1.3 Was ist so toll an Android?

Was erklärt den Erfolg von Android? Rein numerisch gibt es mehr verschiedene Android-Handys als beispielsweise iPhones, nämlich knapp 15.000 – allein schon aufgrund der großen Anzahl verschiedener Hersteller. Diese bieten unterschiedliche Designs und Ausstattungen, sodass jeder sein Lieblingsgerät findet. *Google Play* ist vorinstalliert, und dort bekommen Sie eine siebenstellige Zahl an Apps aus einer Hand, die meisten sogar kostenlos. Sie müssen keine Apps aus obskuren Quellen installieren, wie das noch in der Zeit vor den Smartphones der Fall war – ohne es zu merken, hatten Sie da schnell mal ein Abo für 2,99 € die Woche am Hals, obwohl Sie doch nur einen einzigen Klingelton haben wollten.

Die Apps in *Google Play* werden allerdings nicht von einer Prüfinstanz abgenickt, wie es beim App Store von Apple der Fall ist. Was einerseits ein Vorteil ist, birgt andererseits leider die Gefahr, dass Ihnen eine böswillige App unterkommt. Da hilft es nur, sich genau zu überlegen, welche Befugnisse Sie einer App erteilen – bei der Installation werden diese angezeigt, und wenn sie Ihnen zu freizügig erscheinen, verzichten Sie lieber auf die App. Ein simples Memory-Spiel braucht nicht auf die SMS-Funktion Ihres Handys zuzugreifen, und ein einfacher elektronischer Einkaufszettel benötigt wohl kaum Internetzugriff und Ihre genaue Position – es sei denn, er möchte Ihnen passende Reklame

anzeigen. Mein Tipp: Suchen Sie sich eine kostenlose, werbefreie Notizblock-App (z. B. *OI Notepad*). Erfahrungsgemäß aber kümmern sich nur wenige Nutzer um Zugriffsbeschränkungen. Sonst wären Spiele wie das oben genannte Memory deutlich weniger erfolgreich.

Besitzen Sie schon länger ein Android-Gerät, dann kennen Sie sicher bereits eine Menge sinnvoller Apps – lassen Sie mich Ihnen trotzdem eine kleine Auswahl vorstellen, die aus Sicht eines Entwicklers spannende Aspekte aufweist. Alle hier vorgestellten Apps sind kostenlos (aber nicht alle sind werbefrei). Sie finden sie in *Google Play*, indem Sie den jeweiligen Namen ins Suchfeld eingeben.

1.3.1 OsmAnd Karten & Navigation

Sicher kennen Sie *Google Maps* – es ist schließlich auf fast jedem Android-Smartphone vorinstalliert. Diese App hat allerdings einen entscheidenden Nachteil: Sie lädt die jeweils benötigten Kartendaten aus dem Internet herunter, wenn sie gerade benötigt werden (in neueren Versionen können Sie das allerdings umkonfigurieren).

Wenn Sie sich in einem Gebiet ohne Mobilfunkabdeckung befinden oder im Ausland, wo die Roaming-Kosten für den Datenverkehr schon mal die Kosten für Flug und Hotel übersteigen können, sollten Sie *Google Maps* tunlichst nicht aktivieren.

Die Alternative – *OsmAnd Karten & Navigation* – erlaubt es, Karten vorab herunterzuladen und auf der SD-Karte zu speichern. So können Sie die Daten für alle Gegenden, die Sie zu besuchen gedenken, bequem zu Hause im WLAN herunterladen und verbrauchen später vor Ort keinen Cent Mobilfunkkosten.

OsmAnd (siehe Abbildung 1.7) verwendet das Kartenmaterial von *openstreetmap.org* (daher das Buchstabenkürzel OSM), das von einer offenen Community gepflegt wird. Sie werden staunen, was Sie darauf alles finden: Bushaltestellen mit Linieninformationen, Restaurants und Sehenswürdigkeiten. Manchmal entdecken Sie dort sogar Fuß- oder Radwege, die in *Google Maps* fehlen.

Die heruntergeladenen Karten sind dabei keineswegs große Grafikdateien, sondern Geodaten in einem speziellen Format. Sie werden dynamisch auf den Bildschirm gezeichnet, was Sie beim Zoomen sehr gut erkennen können. Dahinter steckt eine ziemlich umfangreiche Programmlogik, zumal die Karten auch noch abhängig von der Himmelsrichtung frei gedreht werden können. Über den eingebauten Kompass wird in einem späteren Kapitel noch zu reden sein – und auf die Kartendaten von OSM komme ich noch mal ausführlich zurück. Sie werden diese Karten sogar in einer eigenen App verwenden.

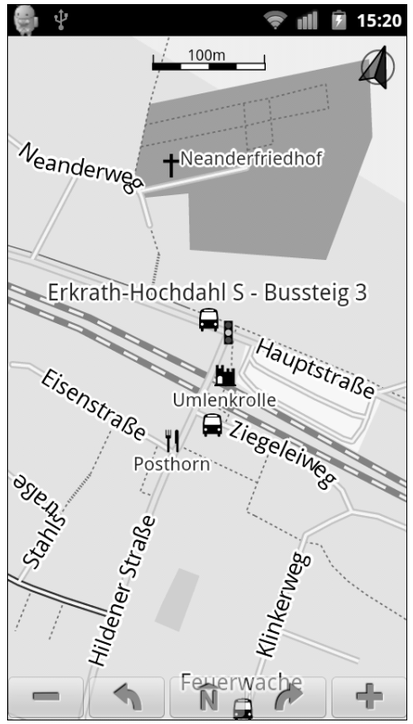

Abbildung 1.7 »OsmAnd« zeigt Ihnen eine Menge Details in der Umgebung – ganz ohne Internetkosten.

1.3.2 Google Sky Map

Während Ihnen *OsmAnd* die Navigation zu Lande erleichtert, benötigen Sie ein virtuelles Planetarium wie *Google Sky Map*, um sich am Himmel zurechtzufinden. Planeten, Sterne, Sternbilder – Sie werden den Himmel mit anderen Augen betrachten, wenn Sie abends mit Ihrem Handy bewaffnet nach oben schauen.

Sie richten das Handy einfach gen Himmel, und dank der Lagesensoren erscheint der passende Himmelsausschnitt, angereichert mit Beschriftungen (siehe Abbildung 1.8). So können Sie sehr schnell herausfinden, ob das helle Licht im Osten ein kräftig leuchtender Stern ist, die Venus (auch bekannt als Abend- bzw. Morgenstern) oder ein Ufo (wenn das Objekt in *Google Sky Map* fehlt).

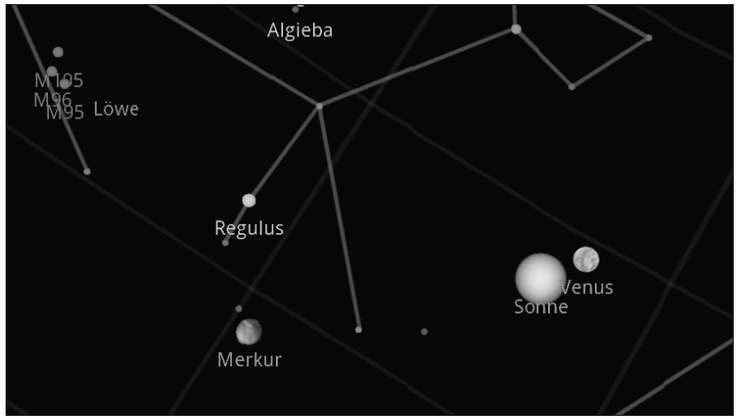

Abbildung 1.8 »Google Sky Map« verrät Ihnen nicht nur die Positionen der Planeten, sondern auch die der Sonne – für den Fall, dass Sie diese mal vor lauter Wolken nicht finden.

Google Sky Map bietet einen Nachtmodus für das an die Dunkelheit gewöhnte Auge (Rot auf Schwarz) und eine Galerie mit Fotos vom Hubble-Teleskop. Leider fehlt die Möglichkeit, ein Objekt anzutippen, um Informationen etwa über Entfernung oder Leuchtkraft zu erhalten – dazu müssen Sie dann doch die Wikipedia (oder eine andere Informationsquelle) bemühen.

Ähnlich einer Karten-App verwendet *Google Sky Map* die Lagesensoren Ihres Geräts und die Geokoordinaten, um den richtigen Himmelsausschnitt anzuzeigen. Dahinter steckt eine Menge Mathematik, die selbst die meisten ausgebildeten Astronomen nicht auswendig programmieren könnten. Deshalb werde ich Ihnen in diesem Buch nur eine einfache Variante des Himmelskugeleffekts erklären.

1.3.3 Hoccer

Es gibt eine ganze Menge Möglichkeiten, um Daten zwischen zwei Handys auszutauschen. Möchten Sie einem Freund Ihre Telefonnummer oder E-Mail-Adresse geben oder ein Foto, das Sie gerade geschossen haben? Oder möchten Sie völlig anonym, aber doch unabhörbar mit einem Bekannten Nachrichten austauschen? Bevor Sie mit Kabeln hantieren oder Ziffernfolgen diktieren, starten Sie doch eine App, und werfen Sie die Daten einfach rüber (siehe Abbildung 1.9).

Wie funktioniert das? Woher weiß *Hoccer*, an welches Handy es Daten übertragen muss? Was ist, wenn zur selben Zeit zwei Leute neben mir »mithören«? Ist das nicht furchtbar unsicher?

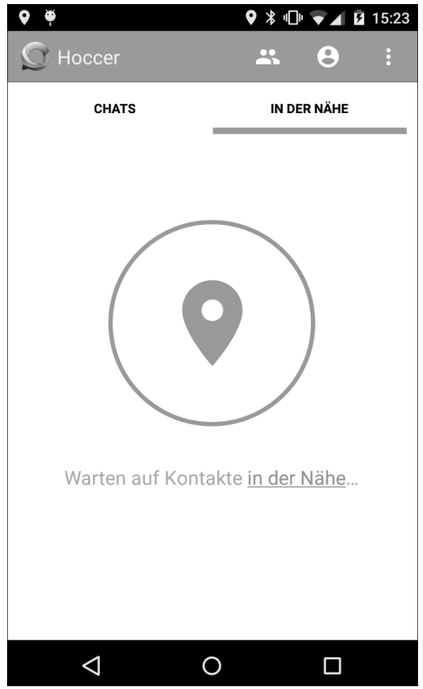

Abbildung 1.9 »Hoccer« schickt Daten per Gestensteuerung von einem Handy zum anderen: Der eine wirft, der andere fängt.

Diese Fragen sind aus Sicht eines Entwicklers sehr spannend. Offensichtlich verwendet die App den Beschleunigungssensor Ihres Handys, um die Gesten zu erkennen. Aber das genügt nicht, denn sobald eine große Anzahl Anwender die App innerhalb eines kurzen Zeitraums verwendet, wäre es unmöglich, zusammengehörende Gesten von zufälligen Übereinstimmungen zu unterscheiden. Deshalb muss die App auf Ihre Geokoordinaten zugreifen. Sobald der Beschleunigungssensor Ihres Handys eine Geste meldet, schickt *Hoccer* die Uhrzeit (auf einige Nanosekunden genau) und Ihren Standort an eine mächtige Internetapplikation. Diese Serveranwendung wertet Geokoordinaten und Zeitpunkte aus und führt passende Paare zusammen. Falls nicht genau zwei Ereignisse zueinander passen, sondern beispielsweise nur eines, drei oder mehr, wird keine Verbindung aufgebaut.

Der Anwender wird dann noch einmal gefragt, ob er wirklich mit der erkannten Gegenstelle Kontakt aufnehmen möchte. Falls ja, kann die eigentliche Datenübertragung beginnen, die dann über eine verschlüsselte Verbindung stattfindet.

Sie sehen, dass *Hoccer* nicht ohne eine mächtige Serveranwendung auskommt. Aber gerade die schnelle, problemlose Verbindung von Smartphone-Sensoren und Rechnern im Internet ist es, die diese (und andere) neuartige Anwendungen ermöglicht. Und weil die Synergie zwischen Smartphone und Internetanwendung so wichtig ist, zeige ich Ihnen noch weitere Apps, die dieses Zusammenspiel nutzen, und erkläre in einem späteren Kapitel genau, wie Sie so etwas selbst bauen können.

1.3.4 c:geo

Es gibt mehrere neuartige Spiele, die das Zusammenwirken von Smartphone-Features und einer Serveranwendung nutzen, und eines der bekanntesten dürfte Geocaching sein. Geocaching ist ein Spiel, das vor Beginn des Smartphone-Zeitalters so gut wie unbekannt war. Es ist nichts anderes als eine Schatzsuche, wobei Ihr Smartphone die Schatzkarte darstellt und Sie mittels Satellitennavigation zum Ziel führt. Dort ist dann unter einem Baumstamm oder in einer Mauerritze ein kleines Kästchen versteckt – der Cache. Darin befinden sich je nach Größe kleine Geschenke, von denen Sie eins gegen ein mitgebrachtes austauschen können. Außerdem gibt es eine Finderliste, in die Sie sich eintragen können. Gleichzeitig können Sie per Smartphone Ihren Fund auf der Webseite von *geocaching.com* bekannt geben (wenn Sie sich dort angemeldet haben). Manchmal finden Sie in der Beschreibung eines Caches ein kleines Rätsel, das Sie lösen müssen, um den genauen Fundort zu identifizieren.

Man mag es kaum glauben, aber die GPS-Schatzsuche lockt sogar notorische Stubenhocker ins Grüne: Achten Sie mal beim nächsten Spaziergang darauf, wie viele Familien mit Smartphones durch den Wald laufen auf der Suche nach dem nächsten Cache. Es gibt eine ganze Reihe Apps, die die Schatzsuche mit dem Handy ermöglichen, und die beliebteste dürfte *c:geo* sein (siehe Abbildung 1.10).

Für Entwickler gibt es eine ganze Reihe interessanter Fragen, die die Programmierung einer solchen App betreffen. Dass sie den GPS-Sensor des Smartphones verwenden muss, um die aktuellen Geokoordinaten zu erhalten, ist offensichtlich. Anschließend muss die App bei einer Serveranwendung anfragen, welche Caches sich in der Umgebung befinden. Spannend ist die Bildschirmanzeige: Dort dient z. B. die von *Google Maps* zur Verfügung gestellte Karte als Grundlage, und für die Caches werden an der richtigen Stelle kleine Grafiken eingeblendet.

Manchmal sind die GPS-Koordinaten nicht genau genug – weniger als 10 m Genauigkeit liefern sie nicht. Wenn Sie einen Cache von 5 cm Größe finden wollen, brauchen Sie Hilfestellung. Dazu blendet *c:geo* auf Wunsch einen Kompass ein, der Sie in die richtige

Richtung leitet. Auf den Kompass werden wir im Laufe dieses Buches noch zurückkommen, und Geokoordinaten dürfen natürlich auch nicht fehlen.

Abbildung 1.10 »c:geo« blendet die nahen Geocaches in einer Karte ein. Wie Sie sehen, befinden sich im Neandertal gleich mehrere Fundorte.

1.3.5 Barcode & QR Scanner

Jedes Smartphone besitzt eine eingebaute Kamera, und jedes Produkt im Regal besitzt einen Barcode. Diese Codes identifizieren Produkte eindeutig, und dank einer umfangreichen Datenbank im Internet können Sie beliebige Informationen nachschlagen.

Eine der Apps, die Ihnen dabei hilft, ist das frühere *barcoo* (siehe Abbildung 1.11), das heute auf den sperrigen Namen *Barcode & QR Scanner barcoo* hört. Scannen Sie den Strichcode eines Produkts, und sofort schickt die App den erkannten Code zu ihrer Serveranwendung, die postwendend alles dazu ausspuckt, was sie weiß. Je nach Art des Produkts erhalten Sie aktuelle Preise, bei Lebensmitteln eine Ampel (größer und farbiger als auf den meisten Produkten) und Informationen zur ökologischen Verträglichkeit. Bei Büchern oder DVDs finden Sie manchmal Rezensionen und Bewertungen.

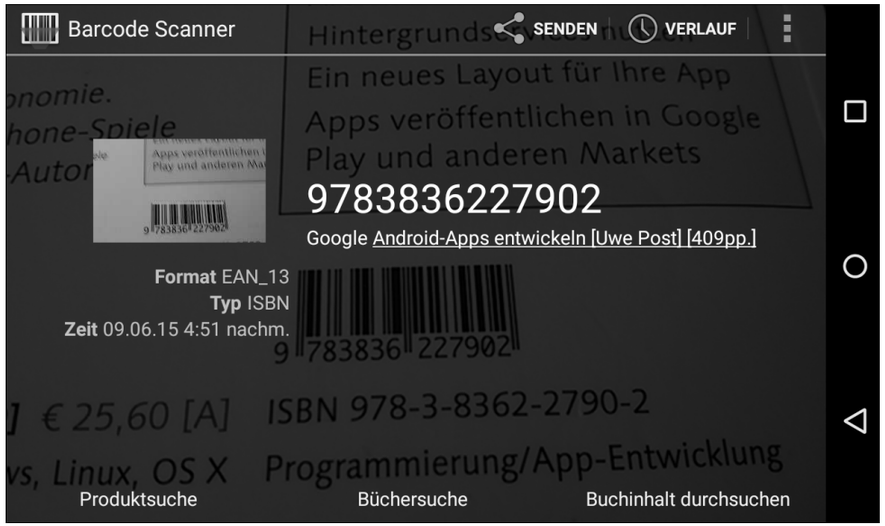

Abbildung 1.11 »Barcode & QR Scanner« scannt Produktcodes und durchsucht das Internet danach.

Wenn Sie diesen Scanner ausprobieren, wird Ihnen sicher auffallen, dass Sie die Kamera nicht auslösen müssen, um einen Barcode zu scannen. Die App arbeitet nämlich mit dem Vorschaubild der eingebauten Digicam. Natürlich haben Vorschaubilder bei Weitem nicht die Auflösung von endgültigen Schnappschüssen, aber sie genügen, um die verhältnismäßig einfachen Muster eines Barcodes zu erkennen. In der App steckt einiges an Programmierarbeit, denn es ist keineswegs trivial, die Balken in einem verrauschten, grobpixeligen Bild richtig zu erkennen. Sie werden in einem späteren Kapitel dieses Buches erfahren, wie das funktioniert.

1.3.6 Öffi

Eine weitere App, die das Zusammenspiel von Ortsdaten, *Google Maps* und einer Serveranwendung nutzt, ist *Öffi*. Wenn Sie viel mit öffentlichen Verkehrsmitteln unterwegs sind, kann Ihnen diese App unschätzbare Dienste erweisen (siehe Abbildung 1.12).

Öffi genügt in den meisten Fällen die netzwerkbasierte Ortsbestimmung, d. h., es kommt ohne GPS aus, das Ihren Akku ziemlich schnell leeren kann. Eine Serveranwendung verrät *Öffi* die Koordinaten der nahe gelegenen Haltestellen mitsamt den dort verkehrenden Linien und Fahrplänen. Die Haltestellen blendet *Öffi* in eine Karte von *Google Maps* ein, ähnlich wie *c:geo* dies mit Geocaches tut.

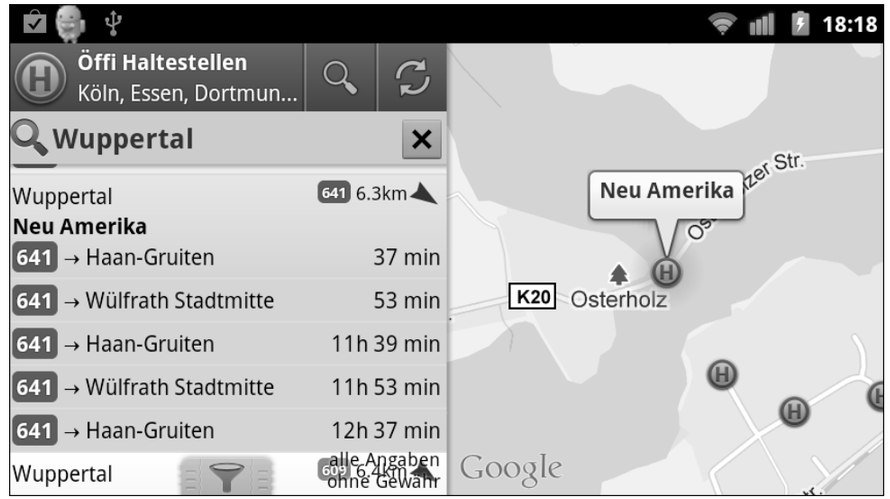

Abbildung 1.12 »Öffi« weiß immer, wann der letzte Bus kommt – und wann der erste.

Hinzu kommt eine Auskunft über die schnellste Verbindung zu einer beliebigen Zielhaltestelle. Wenn Sie im Zug oder Bus sitzen, können Sie sich den Verlauf der Fahrt und gegebenenfalls Informationen zum Umsteigen anschauen. Noch dazu werden aktuelle Verspätungen berücksichtigt.

Es ist offensichtlich, dass der größte Batzen Arbeit hier in der Implementierung der Kommunikation mit den Servern der Verkehrsunternehmen steckt.

1.3.7 Wikitude World Browser

Eine weitere App, die sich die Synergie zwischen Ortsbestimmung und einer Internetdatenbank zunutze macht, ist *Wikitude*. Genau genommen greift *Wikitude* sogar auf eine ganze Reihe von Datenquellen zu – so ungefähr auf alles, was mit Geokoordinaten versehen ist. Alles, was *Wikitude* in der Nähe findet, wird dann in Form von kleinen Fähnchen ins Kamerabild eingeblendet (siehe Abbildung 1.13).

Freilich ist nicht alles besonders sinnvoll: Beispielsweise ist es meist von geringem Interesse, sich Fotos von dem Ort anzusehen, an dem Sie ohnehin gerade stehen. Doch wer weiß, vielleicht ergibt sich eine überraschende Perspektive, wenn Sie sich anschauen, wie die Benutzer von *flickr.com* Ihren Standort in Szene gesetzt haben.

Tendenziell lustig ist die Anzeige nahe gelegener Internet-Webcams: Theoretisch könnten Sie sich in deren Erfassungsbereich stellen und sich auf diese Weise selbst zuwinken.

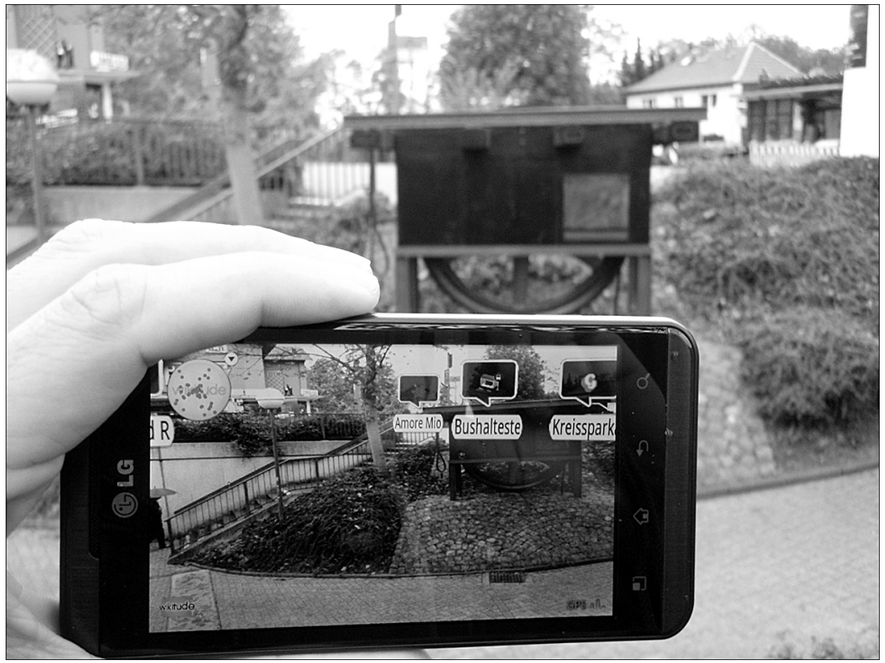

Abbildung 1.13 »Wikitude« zeigt Ihnen, wo der Bus hält, selbst wenn ein Gegenstand den Blick versperrt.

Deutlich nützlicher sind Fähnchen, die zur nächsten Pizzeria, Apotheke oder Übernachtungsmöglichkeit führen. Wenn Sie in der Fremde unterwegs sind, können Sie an Sehenswürdigkeiten sofort den zugehörigen Wikipedia-Artikel aufrufen.

Ähnlich wie *Öffi* verwendet *Wikitude* Ihre Geokoordinaten, um auf dem Server eine Umgebungssuche durchzuführen. Die Spezialität ist das Einblenden von Hinweisfähnchen ins Vorschaubild der Kamera. Dabei ist es allerdings keineswegs so, dass *Wikitude* Objekte im Kamerabild erkennt (dessen können Sie sich vergewissern, indem Sie einfach das Objektiv mit dem Finger verdecken). Vielmehr ermittelt die App den Winkel, in dem Sie Ihr Handy halten, und berechnet dann, welche Fähnchen an welcher Stelle auf dem Bildschirm einzublenden sind.

Diese Vermischung von Kameravorschau und hineinplatzierten zusätzlichen Inhalten, die sogenannte *Augmented Reality*, ist die Königsdisziplin der Smartphone-Entwicklung. In einem späteren Kapitel werden wir darauf zurückkommen: Dieses spannende Konzept möchte ich Ihnen keinesfalls vorenthalten.

1.3.8 Sprachsuche

Haben Sie sich auch schon einmal darüber geärgert, dass Computer heutzutage dermaßen neunmalklug sind, dass sie Ihnen alles über das Wetter am anderen Ende der Welt verraten können, aber erst nachdem Sie ihnen mit der Tastatur oder Maus die entsprechende Frage gestellt haben? Wo sind die sprechenden Computer, die schon in der ersten Serie von »Raumschiff Enterprise« vorkamen, die jedes Wort verstanden, Kommandos interpretierten und sogar wussten, wann sie *nicht* gemeint waren? Gerade ein Smartphone, das meist auf eine Tastatur verzichten muss, ist überaus unpraktisch, wenn Sie ein paar Sätze eingeben möchten – und das umso mehr, je länger der Text ist und je stärker das Transportmittel, in dem Sie sich gerade befinden, wackelt. Wir brauchen die Spracheingabe!

Nun beherrschen selbst manche Telefone der Vor-Smartphone-Ära schon Sprachwahl, das heißt, Sie sprechen dem Telefon einen Namen vor, und wenn Sie diesen später wiederholen, erkennt das Gerät den Klang der Wörter wieder und wählt die zugehörige Nummer.

Die Google-Sprachsuche geht ein Stück weiter: Hier müssen Sie dem Gerät nicht erst jedes Wort beibringen – das wäre auch etwas viel verlangt. Stattdessen ermittelt die Sprachsuche auf dem Smartphone gewisse Charakteristika Ihrer gesprochenen Wörter und versucht, sie mithilfe einer serverseitigen Datenbank zu identifizieren. Diese Datenbank ist verdammt mächtig, filtert sogar Nebengeräusche aus und kann aus ihren Fehlern lernen. Wenn Sie Ihren Suchbegriff ohne allzu starken Dialekt direkt ins Mikrofon sprechen, funktioniert die Suche erstaunlich gut.

Schon seit Android 2.1 gibt es den recht unscheinbaren Mikrofon-Button auf der virtuellen Tastatur. Egal in welcher App Sie sich befinden – E-Mail, SMS, Notizblock –, tippen Sie das Mikrofon an, und sprechen Sie (siehe Abbildung 1.14).

In der Google-eigenen Suche genügt es auch, die Zauberworte »Okay Google« zu sprechen. Wenn Sie deutlich artikuliert Hochdeutsch reden und keine außergewöhnlichen Wörter verwenden, wird die Qualität der Spracherkennung Sie überraschen.

Die Datenmenge, mit der die Google-Server bei der Sprachsuche umgehen müssen, übersteigt die Vorstellungskraft der meisten App-Entwickler. Das macht aber nichts, denn entscheidend ist: Sie können die Spracherkennung leicht in Ihre App integrieren! Das gilt nicht nur für alle Texteingabefelder, die die Bildschirmtastatur verwenden (dort ist die Spracheingabe automatisch verfügbar), sondern auch für jegliche Anwendung, die Sie sich vorstellen können. Alles ist denkbar, etwa ein Abenteuer-Rollenspiel, in dem Sie zur Wirtin marschieren und lautstark »Bier!« verlangen; wenn Sie Pech haben, antwortet sie: »So was wie dich bedienen wir hier nicht« – in gesprochenen Wörtern, denn eine Sprach-

ausgabe beherrscht Android natürlich auch. Darauf werden wir bereits bei der ersten Beispiel-App zurückkommen, die Sie mit diesem Buch programmieren werden.

Abbildung 1.14 Jede Notizblock-App wird ab Android 2.1 zum Einkaufszettel mit Spracheingabe.

1.3.9 Cut the Rope

Stellvertretend für das auf Smartphones besonders beliebte Genre der sogenannten Physikspiele sei hier *Cut the Rope* genannt. In diesem Spiel müssen Sie eine Süßigkeit in das hungrige Maul eines knuffigen grünen Haustiers befördern. Knifflig daran ist, dass die Leckerei allen möglichen Kräften ausgesetzt ist: der Schwerkraft, dem Auftrieb einer Seifenblase oder gummiartigen Schnüren, die Sie mit dem Finger durchtrennen können (siehe Abbildung 1.15).

Der Clou an Physikspielen wie diesem ist, dass sie praktisch intuitiv spielbar sind, weil jeder aus seiner täglichen Lebenserfahrung die Spielregeln kennt. Das A und O ist natürlich eine akkurate Simulation der physikalischen Kräfte. Den Job erledigt eine Physik-Engine, die alle Kräfte kennt, die auf Spielobjekte wirken, und daraus die natürliche Bewegung errechnet.

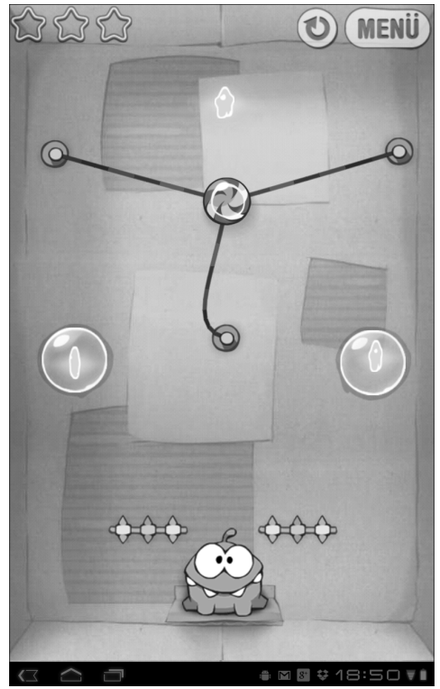

Abbildung 1.15 »Om Nom« steht auf Süßigkeiten. Der Fütterung des knuffigen Wesens stehen allerdings knifflige Physikrätsel im Wege.

Diese Regeln und Formeln sind überraschend einfach, allerdings hilft es bei der Programmierung, ein bis zwei Semester Physik studiert oder zumindest in der Schule im Physikunterricht gut aufgepasst zu haben. Für Mathe-Muffel gibt es fertige Physik-Engines, die bereits mit einer mächtigen Grafik-Engine gebündelt sind, z. B. die für Privatpersonen frei verfügbare *Unity3D Personal Edition* (*unity3d.com*). Im einfachsten Fall müssen Sie nur ein paar geometrische Objekte positionieren, der Engine unterschieben und ihr mitteilen, welche Kräfte wirken sollen.

1.3.10 Shaky Tower

Ein weiteres Physikspiel, das sogar die Sensoren des Smartphones nutzt, ist *Shaky Tower*. Hier müssen Sie Aufgaben lösen, die meist damit zu tun haben, dass Sie grinsende Quadrate aufeinandersetzen (siehe Abbildung 1.16). Dabei spielt die Schwerkraft eine entscheidende Rolle: Ein schiefer Turm kippt unweigerlich um. Wie schon bei *Cut the Rope*

werkelt hier im Hintergrund eine Physik-Engine, die sich um die Berechnung der Kräfte und der Bewegungen kümmert.

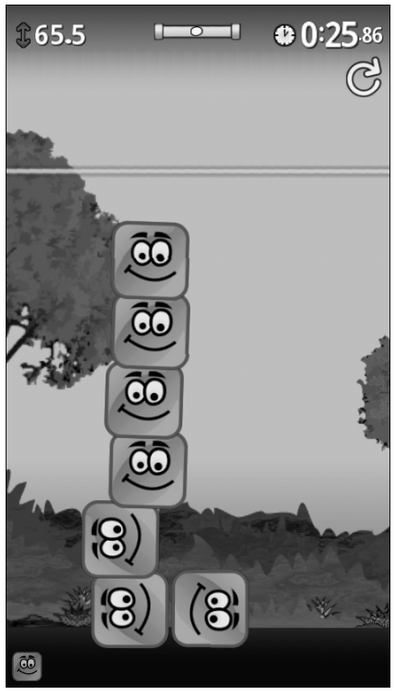

Abbildung 1.16 »Shaky Tower« lässt Sie Türmchen bauen, die so lange stehen bleiben, bis Sie das Handy schräg halten.

Der Unterschied: Nicht nur die Physik des virtuellen Raums wird berücksichtigt, sondern auch die reale. Die Lagesensoren des Handys werden ausgelesen und fügen dem Geschehen auf dem Bildschirm weitere Gravitationskräfte hinzu.

Es gibt eine ganze Reihe von Spielen, die auf dieses Konzept setzen, allerdings verbraucht sich der Effekt recht schnell. Technisch ist interessant, dass Sie der verwendeten Physik-Engine äußere Kräfte hinzufügen müssen, die sich aus den Werten der Lagesensoren berechnen. Die größte Herausforderung besteht wie bei den meisten Spielen im Balancing: Wenn die Bewegung des Handys zu starken Einfluss auf das Spielgeschehen hat, wird es viele Spieler überfordern. Ist der Einfluss zu gering, wird der Effekt als zu schwach empfunden. Hier ist das Fingerspitzengefühl des Programmierers gefragt – und jede Menge Testen.

Kapitel 3
Vorbereitungen

»Ja, ich habe die Lizenzbedingungen gelesen.«
(Häufigste Lüge des 21. Jahrhunderts)

Alle Theorie ist grau, rote Tomaten hin oder her. Sobald Sie Ihren Rechner vorbereitet haben, können Sie mit der ersten App beginnen. Also ran an die Installation!

3.1 Was brauche ich, um zu beginnen?

Wenn Sie einen PC mit Internetanschluss besitzen, können Sie sofort loslegen. Allerdings sollte Ihr PC wenigstens 4 GB RAM haben und eine mit etwa 2 GHz getaktete CPU. Weniger geht auch, allerdings können dann hier und da Wartezeiten auftreten. Mehr ist immer besser!

Sorgen Sie dafür, dass auf Ihrer Festplatte ein paar Gigabyte frei sind. Speicherplatz ist heutzutage so billig, dass ich keine Rücksicht darauf nehmen werde, wie viel davon Sie auf dem Entwicklungsrechner verbrauchen. Nach und nach werden sich eine Menge kleine Dateien ansammeln, und niemand möchte Zeit damit verschwenden, sie einzeln aufzuräumen. Eine SSD ist empfehlenswert.

Um selbst entwickelte Apps auszuprobieren, benötigen Sie ein Android-Gerät (siehe Abbildung 3.1). Prinzipiell kommt jedes Tablet oder Smartphone infrage, auf dem Android 4.0.4 oder neuer läuft. Obligatorisch ist ein USB-Kabel, damit Ihr PC sich mit dem Androiden unterhalten kann. Glücklicherweise liefern die meisten Hersteller solche Kabel mit. Falls nicht, ist das nicht weiter tragisch: Mit wenigen Ausnahmen verfügen Android-Geräte über standardisierte Mikro-USB-Buchsen. Passende Kabel bekommen Sie preiswert in vielen Geschäften oder in Onlineshops.

USB-Treiber

Unter Windows müssen Sie möglicherweise einen USB-Treiber Ihres Handyherstellers installieren, damit die Kommunikation klappt.

Für Google-Telefone wie die Nexus-Geräte finden Sie den Treiber hier:
http://developer.android.com/sdk/win-usb.html

Bei anderen Herstellern hilft ein Blick auf deren Homepage. Freundlicherweise hat Google eine Liste mit passenden Links zusammengestellt:
https://developer.android.com/studio/run/oem-usb.html

Abbildung 3.1 Suchen Sie sich aus, mit welchem Android-Gerät Sie Ihre ersten Apps basteln: Hier sehen Sie ein Tablet und zwei ältere Handys.

Sie benötigen nicht das neueste und schnellste Android-Gerät. Zur Entwicklung für mobile Endgeräte gehört nämlich eine wichtige Erkenntnis: Selbst moderne Smartphones sind nicht so gut bestückt wie ein Desktop-PC oder Notebook. Während ein PC über praktisch unendlich viel Speicher und Plattenplatz verfügt, gilt das für ein Handy nicht. Dort sind fette und langsame Applikationen sogar ein echtes Ärgernis – und werden schnell wieder deinstalliert. Würden Sie eine App entwickeln, die auf einem schnellen Gerät »einigermaßen« läuft, wäre sie auf einem älteren unerträglich – und würde schneller wieder gelöscht, als Sie »aber auf meinem …« sagen können. Daher werden wir von Anfang an darauf achten, möglichst schlanke und schnelle Anwendungen zu schreiben.

Notfalls können Sie auch ohne Android-Gerät loslegen. Es existiert ein Emulator, den Sie auf dem PC oder Mac laufen lassen können. Er simuliert ein Android-System nach Wahl, läuft allerdings auf schmalbrüstigen PCs je nach Anwendung quälend langsam. Ein 64-Bit-Betriebssystem wird dringend empfohlen. Da es für die Qualität einer App entscheidend ist, wie sie sich auf dem Handy bedienen lässt, ist der Emulator immer zweite Wahl bei der Entwicklung.

Für die entspannte Arbeit empfehle ich Ihnen darüber hinaus eine Kaffeemaschine oder einen Wasserkocher mitsamt opulentem Schwarzteevorrat. Übertriebene Taktfrequenzen sind dabei verzichtbar, aber Sie werden staunen, wie hilfreich diese Geräte sind.

Bei der Entwicklung wird uns das *Android Studio* unterstützen, das Anfang 2015 die Nachfolge des Eclipse ADT Bundle angetreten hat. Es basiert auf der bewährten Entwicklungsumgebung *IntelliJ IDEA*. Zum Glück wird hinsichtlich der Betriebssystemvoraussetzungen niemand ausgeschlossen: Das Programm läuft unter Windows, Linux und OS X und sieht auf allen Systemen weitgehend identisch aus.

Sie finden alle Pakete im Download-Angebot (*www.rheinwerk-verlag.de/4586*) im Ordner *software*. (Verwenden Sie die 64-Bit-Version, wenn Ihr System diese unterstützt.) Etwaige neuere Versionen können Sie aber auch kostenlos direkt bei Google herunterladen:

http://developer.android.com/sdk/index.html

Da es sich um erhebliche Datenmengen handelt, sollten Sie für das Herunterladen etwas Zeit einplanen.

3.2 Schritt 1: Android Studio installieren

In der Vergangenheit mussten Entwickler mühevoll Eclipse, ein Java Development Kit (JDK), das Android SDK und Plug-ins separat installieren. Diese Zeiten sind vorbei, seitdem Google das Android Studio anbietet. Seit Version 2.2 enthält Android Studio ein eingebautes *OpenJDK*, sodass Sie nur noch die Entwicklungsumgebung installieren müssen.

Das Rundum-sorglos-Paket besteht aus einem ZIP-Archiv von erheblicher Größe (knapp 800 MB). Dafür verkürzt es den Installationsprozess auf ein paar Klicks. Holen Sie sich das für Ihr System passende Archiv von *https://developer.android.com/studio* und folgen Sie der integrierten Installationsanleitung. Starten Sie die Studioanwendung über das Start-Menü oder das Desktop-Icon.

Eine Entwicklungsumgebung wie AS vereint alle benötigten Funktionen in einem großen Fenster und unterstützt Sie mit praktischen Hilfen beim Programmieren.

AS ist modular angelegt, sodass Sie je nach Bedarf Plug-ins hinzufügen können. Es gibt sogar Plug-ins für andere Programmiersprachen.

Die Screenshots in diesem Buch stammen übrigens von Android Studio unter Linux; das Programm sieht aber auf jedem System nahezu identisch aus.

Abbildung 3.2 Wählen Sie beim ersten Start die »Standard«-Installation aus.

Beim ersten Start ermöglicht AS Ihnen, Einstellungen aus einer vorherigen Installation zu importieren. Da Sie natürlich keine solche Installation haben, klicken Sie einfach auf OK. Wählen Sie im nächsten Schritt die voreingestellte STANDARD-Variante für Ihr Setup und danach ein UI-Theme Ihrer Wahl.

Anschließend möchte AS wissen, ob Sie Spezialwünsche haben (siehe Abbildung 3.2). Für die Beispiele in diesem Buch genügt es, die Installation im Standardmodus durchlaufen zu lassen. Im Anschluss können Sie auf Wunsch das Erscheinungsbild der Anwendung verändern. Wichtiger ist jedoch der Download des SDK.

3.3 Schritt 2: Das Android SDK

Ohne das Android SDK geht nichts. SDK steht für *Software Development Kit*. Dahinter verbirgt sich ein Bündel Programme und Dateien, die die Entwicklung von Android-Apps ermöglichen.

Android Studio zeigt Ihnen im letzten Schritt der Installation, welche zusätzlichen Komponenten nachzuladen sind. Im Standardfall ist das jeweils die neueste Version der SDK-Komponenten. Sie können später mit dem Android SDK Manager weitere Pakete oder andere Versionen nachladen, aber für den Anfang folgen Sie der Empfehlung von Android Studio (siehe Abbildung 3.3).

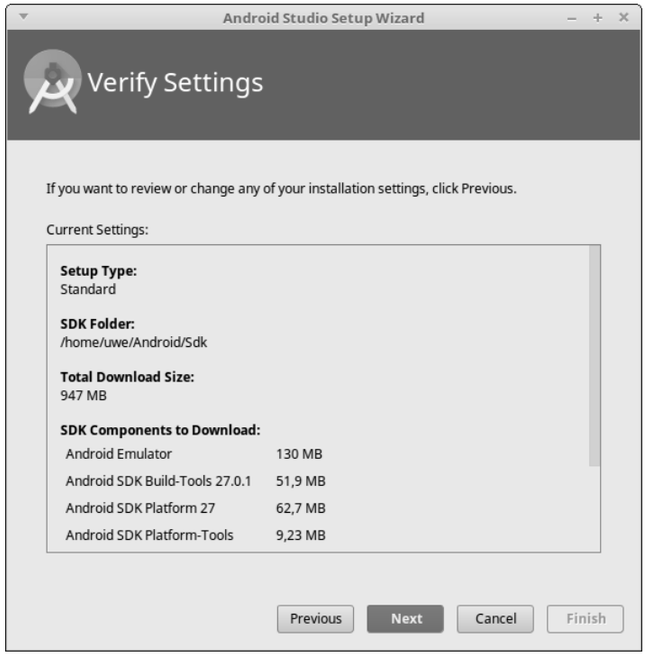

Abbildung 3.3 Android Studio lädt automatisch die wichtigsten Komponenten nach.

Der Download wird je nach Bandbreite Ihrer Internetverbindung eine Weile dauern.

Später wird AS Sie benachrichtigen, wenn Google neue Versionen bestimmter Komponenten zur Verfügung gestellt hat. Sie können solche Hinweise normalerweise ignorieren, es sei denn, Sie legen Wert darauf, immer auf dem aktuellen Stand zu sein. Stellen Sie dazu sicher, dass im SDK Manager (Menü: TOOLS • ANDROID • SDK MANAGER) nirgendwo UPDATE AVAILABLE steht. Falls doch, installieren Sie die neuen Versionen. Löschen Sie ruhig von Zeit zu Zeit ältere Versionen, um Platz auf der Festplatte zu schaffen.

Der SDK Manager bietet Ihnen nicht nur immer die neuesten Versionen der SDK Tools, sondern auch Entwickerunterstützung für nagelneue Techniken wie *Android TV* und

Android Wear. Die besprechen wir zwar nicht alle in diesem Buch, aber falls Ihnen später mal langweilig ist, beschaffen Sie sich einfach alles Gewünschte über diesen Dialog.

Android-Versionen

Wir verwenden Android 8.1 als Basis-SDK, weil wir damit alle modernen Features unterstützen, selbst wenn wir sie nicht verwenden. Die folgende Tabelle zeigt Ihnen die wichtigsten Versionen und die zugehörigen API-Level:

Versionsnummer	Codename	API-Level	Erscheinungsdatum
Android 1.0	»Base«	API 1	2008, veraltet
Android 1.5	»Cupcake«	API 3	2009, veraltet
Android 1.6	»Donut«	API 4	2009, veraltet
Android 2.0	»Eclair«	API 5	2009
Android 2.2	»Froyo«	API 8	2010
Android 2.3	»Gingerbread«	API 9	2010
Android 3.0	»Honeycomb«	API 11	2011, Tablet-Zweig
Android 4.0	»Ice Cream Sandwich«	API 14	2011
Android 4.1	»Jelly Bean«	API 16	2012
Android 4.4	»KitKat«	API 19	2013
Android 5.0	»Lollipop«	API 20	Oktober 2014
Android 5.1.1	»Lollipop_MR1«	API 22	März 2015
Android 6.0	»Marshmallow«	API 23	Oktober 2015
Android 7.0	»Nougat«	API 24	August 2016
Android 8.0	»Oreo«	API 26	August 2017
Android 8.1		API 27	Oktober 2017

Tabelle 3.1 Die Android-Versionen und ihre API-Level

Vor den Downloads müssen Sie gegebenenfalls Lizenzbedingungen akzeptieren, dann erst erfolgt die Installation. Die Daten landen übrigens in einem Ordner *Android* in Ihrem Nutzerverzeichnis. Falls Sie im SDK Manager ein Emulator-Systemimage zum Download ausgewählt haben, bereitet der Wizard ein virtuelles Gerät vor, auf dem Sie später Apps starten können.

Zum Schluss heißt AS Sie freundlich willkommen (siehe Abbildung 3.4).

Abbildung 3.4 Android Studio wartet auf Ihr erstes Projekt.

Von hier aus erreichen Sie bei Bedarf erneut den SDK Manager (über das Drop-down CONFIGURE). Viel interessanter ist aber natürlich die erste Option START A NEW ANDROID STUDIO PROJECT: Sie erstellt ein neues Android-Projekt. Also, los geht's!

3.4 Ein neues App-Projekt anlegen

Der Dialog CREATE ANDROID PROJECT verlangt zunächst einen Namen für die App (siehe Abbildung 3.5). Denken Sie sich etwas aus oder belassen Sie es bei *My Application*. Als Firmenname verwende ich in diesem Buch die (nicht existierende) Domain *android-newcomer.de*. Natürlich können Sie auch etwas anderes wählen. Entscheidend ist die Schreibweise in Kleinbuchstaben und mit trennenden Punkten: Der Wizard leitet aus Ihrer Angabe automatisch den PACKAGE NAME für das Projekt ab, indem er die Reihenfolge der Elemente umdreht (also `de.androidnewcomer`).

Die PROJECT LOCATION müssen Sie nicht ändern, es sei denn, Sie verwenden ein anderes als das vorgegebene Basisverzeichnis.

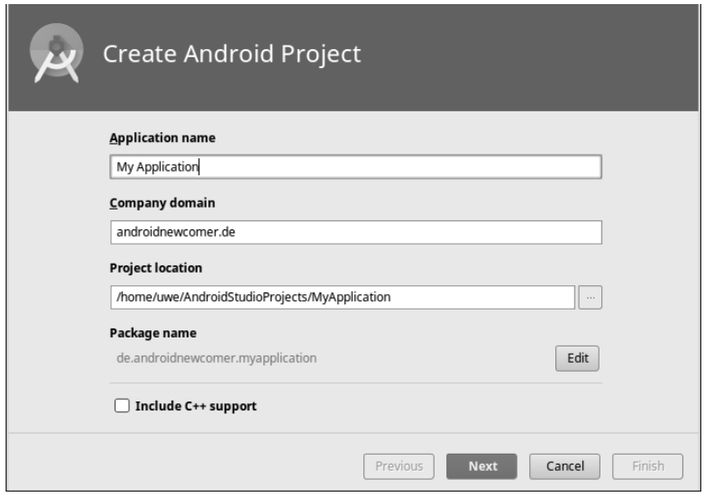

Abbildung 3.5 Konfigurieren Sie Ihr erstes Testprojekt.

Der Wizard möchte im nächsten Schritt von Ihnen wissen, auf welchen Geräten Ihr Projekt laufen soll (siehe Abbildung 3.6).

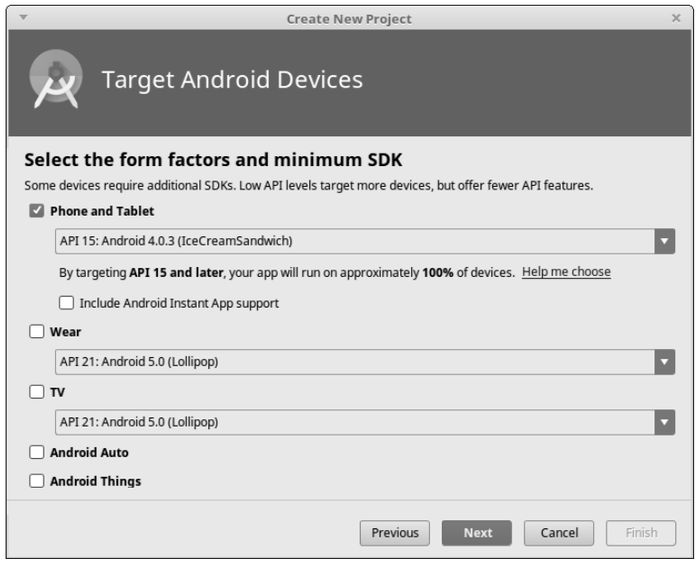

Abbildung 3.6 Wählen Sie aus, auf welchen Geräten Ihre App laufen soll.

Setzen Sie den Haken nur bei PHONE AND TABLET, und stellen Sie als MINIMUM SDK zum Beispiel API 15: ANDROID 4.0.3 ein. Dies ist die älteste Android-Version, auf der Ihre App funktionieren wird. Auf noch älteren Geräten würde die Installation fehlschlagen.

Freundlicherweise zeigt der Wizard Ihnen gleich, wie viel Prozent der weltweiten Android-Geräte Sie mit Ihrer Auswahl unterstützen. Im Fall von Android 4.0.3 wären das 100 %, da ältere Versionen inzwischen praktisch ausgestorben sind. Da die Beispiele in diesem Buch keine moderneren Features von Android voraussetzen, sind Sie mit dieser Einstellung gut bedient.

Im nächsten Schritt bietet der Wizard Ihnen an, vorgefertigte Komponenten in die neue App einzufügen. Verzichten Sie darauf, indem Sie EMPTY ACTIVITY auswählen (siehe Abbildung 3.7).

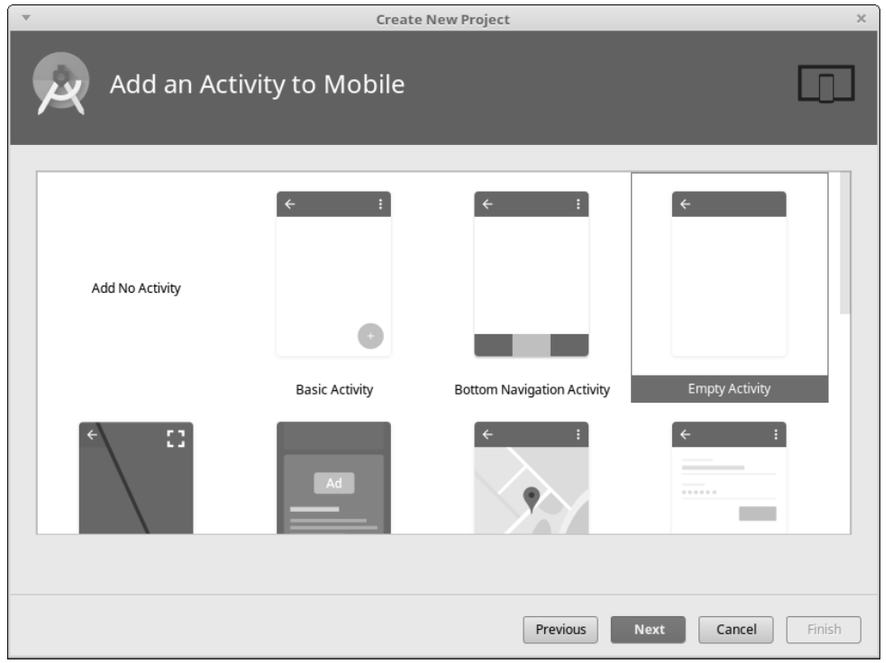

Abbildung 3.7 Für den ersten Testlauf benötigen Sie keine Zusätze. Wählen Sie »Empty Activity«.

Android-Apps bestehen gewöhnlich aus *Activities*. Was das genau bedeutet, erkläre ich Ihnen später. Schalten Sie im letzten Wizard-Schritt den Kompatibilitätsmodus aus, was den Code übersichtlicher macht. Dann klicken Sie auf FINISH (siehe Abbildung 3.8).

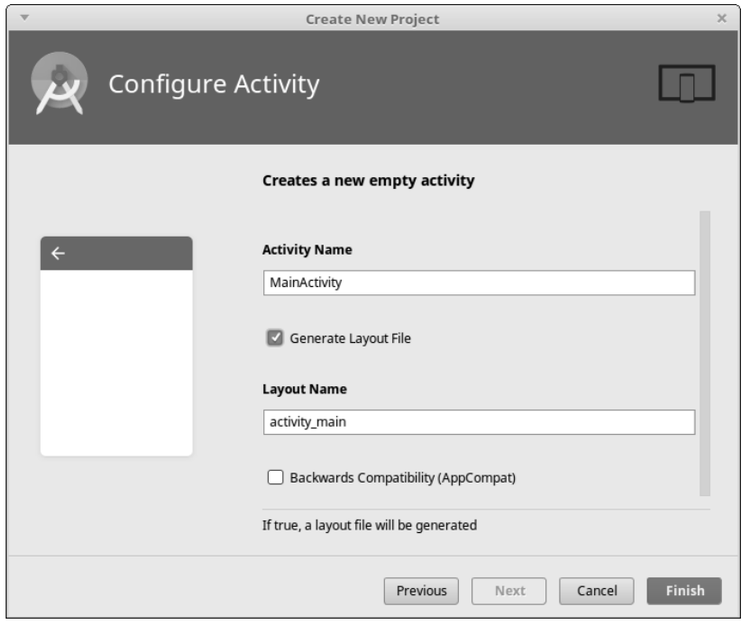

Abbildung 3.8 Die vorgegebenen Bezeichnungen im letzten Wizard-Schritt übernehmen Sie einfach – entfernen Sie lediglich den Haken bei »Backwards Compatibility«.

Endlich erzeugt Android Studio für Sie das neue leere Projekt. Da hierzu erneut Daten heruntergeladen werden müssen, können Sie während der Wartezeit zu Ihrem Handy greifen.

Durcheinander mit Build-Skript und SDK-Versionen beseitigen

Möglicherweise präsentiert sich das frisch erzeugte App-Projekt zunächst mit einem Fehler der Sorte »Failed to find target ...«. In diesem Fall starten Sie zunächst sicherheitshalber die Entwicklungsumgebung über den Menüpunkt File • Invalidate Caches and Restart neu.

Wenn auch das nicht hilft, prüfen Sie zunächst, welche SDK-Version installiert ist. Öffnen Sie den SDK Manager (das Icon mit dem blauen Pfeil und dem grünen Roboterkopf), und schauen Sie, welche Versionsnummer angehakt ist. Falls mehrere SDKs installiert sind, merken Sie sich die höchste Versionsnummer (z. B. 27).

Dann öffnen Sie die Datei *app/build.gradle* (zu finden links im Projektbaum – nicht zu verwechseln mit der gleichnamigen Datei im Wurzelverzeichnis Ihres Projekts!) und stellen sicher, dass hinter `compileSdkVersion` und `targetSdkVersion` die Versionsnummer des installierten SDK steht (z. B. 27). Der Wizard erzeugt hier manchmal eine zu kleine Nummer und referenziert so ein nicht installiertes SDK. Nach der Änderung klicken Sie auf Try Again am oberen Rand des Fensters.

Ein ganz ähnliches Problem signalisiert die Fehlermeldung »Failed to find Build Tools revision ...«. Welche Build Tools installiert sind, erfahren Sie, wenn Sie im SDK Manager auf der Registerkarte SDK Tools unten rechts die Package Details einblenden. Ein Haken könnte bei einer frischen Installation beispielsweise bei Version 27.0.1 gesetzt sein.

Im Build-Skript app/build.gradle erzeugt der Wizard oft einen falschen Eintrag `buildToolsVersion` (oder gar keinen). Sorgen Sie dafür, dass die aktuellste installierte Version referenziert wird: buildToolsVersion '27.0.1'. Klicken Sie dann erneut auf Try Again.

3.5 Android Studio mit dem Handy verbinden

Um Android Studio mit Ihrem Handy zu verbinden, öffnen Sie zunächst dessen *Einstellungen*-App über das Hauptmenü. Suchen Sie dann den Menüpunkt Entwickleroptionen. Schalten Sie das USB-Debugging an, indem Sie den Schalter aktivieren. Sorgen Sie mit Aktiv lassen außerdem dafür, dass das Handy nicht einschläft, während es am USB-Kabel hängt (siehe Abbildung 3.9).

Falls Sie die Entwickleroptionen nicht finden können, müssen Sie Ihr Handy zunächst davon überzeugen, dass Sie qualifiziert sind, diese zu bearbeiten. Dazu suchen Sie den Menüpunkt Über das Telefon. Am Ende der Liste finden Sie die Build-Nummer. Tippen Sie diese siebenmal an, und Sie haben die Entwickleroptionen freigeschaltet.

Schließen Sie als Nächstes Ihr Handy mit einem passenden Kabel an eine USB-Schnittstelle Ihres Rechners an. Je nach Android-Version müssen Sie anschließend noch einmal die Verbindung bestätigen oder die Verwendung durch den Computer erlauben (am besten permanent). Ist die Verbindung einmal hergestellt, erscheint Ihr Gerät innerhalb von Android Studio im Logcat-Fenster. Darüber können Sie viele Funktionen des Geräts fernsteuern. Dazu kommen wir aber später.

Vor allem können Sie Ihre App mit einem Klick auf den grünen Pfeil in der Icon-Leiste starten. Beim ersten Mal dauert das eine ganze Weile, und das Resultat ist nicht besonders zeigenswert. Aber immerhin: Android Studio baut die App, installiert sie auf Ihrem Handy und startet sie dort. Mehr noch: Sie werden bei Bedarf den Programmablauf Schritt für Schritt verfolgen können, was bei der Fehlersuche ungemein hilfreich ist.

Und Fehlersuche – so traurig es klingt – ist eine der Arbeiten, mit denen Programmierer deutlich mehr Zeit verbringen als mit ihren Freundinnen.

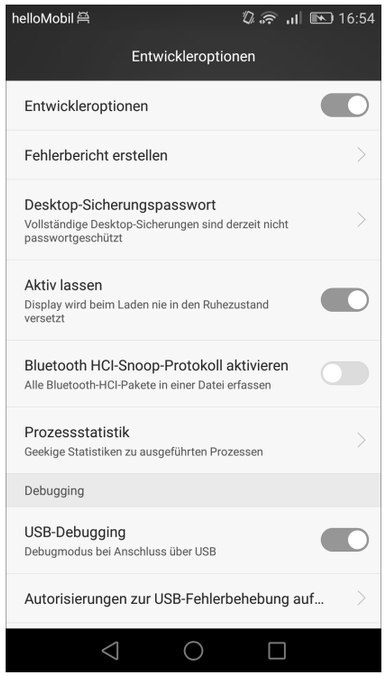

Abbildung 3.9 »USB-Debugging« vereinfacht das Entwicklerleben.

Instant Run

Seit Android Studio 2.0 existiert ein Feature namens *Instant Run*.

Dahinter verbirgt sich ein Mechanismus, der das Entwickeln einer App deutlich beschleunigt: Wann immer es möglich ist, transportiert AS nämlich nur die veränderten Codeteile zum Handy, nicht das ganze APK (Android Application Package). Gerade bei großen Anwendungen ist der Unterschied spürbar.

Voraussetzungen sind Gradle 2.0.0 und eine minSdkVersion 15 oder höher. Je höher die minSdkVersion, in desto mehr Fällen kann AS Instant Run verwenden. Kommen Sie aber nicht auf die Idee, nur wegen Instant Run Ihre App auf Android SDK 21 oder neuer zu beschränken, bloß weil Sie dann schneller entwickeln können. Spätestens wenn Sie eine Release-Version bauen, sollten Sie überlegen, die minSdkVersion so weit herunterzusetzen wie möglich, um möglichst viele Endgeräte zu unterstützen.

3.6 Fehlersuche

Es ist genauso erfreulich wie selten, dass eine App auf Anhieb funktioniert. Was tun, wenn etwas schiefgeht?

Es kommt drauf an, *was* schiefgeht. Jeder Fehler bedeutet zunächst einmal einen Forschungsauftrag an den Entwickler. Das ähnelt der Frage, warum Vögel Federn haben. Um der Antwort auf die Spur zu kommen, kann man Fossilien ausgraben und untersuchen oder einen Schöpfer postulieren. Ich versichere Ihnen, dass die erstgenannte Variante bei der Suche nach Fehlern in Android-Apps zielführender ist als die zweite.

Sie erleben zunächst einmal nur einen *Effekt* des Fehlers – bei Android nicht selten in Form eines Dialogs: »Tut uns leid!« (siehe Abbildung 3.10).

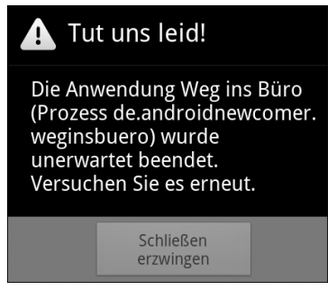

Abbildung 3.10 Ein Fehler ist aufgetreten, und eines ist ziemlich sicher: Es einfach erneut zu versuchen, wird nicht helfen.

Schlüpfen Sie in die Rolle von Sherlock Holmes, um der Ursache des Fehlers auf die Spur zu kommen.

3.6.1 Einen Stacktrace lesen

Wenn Sie Ihr Smartphone mit Android Studio verbunden haben oder mit dem Emulator arbeiten, können Sie sehr genau verfolgen, was die ganze Zeit auf dem Handy passiert. Dazu dient die View LOGCAT. Aktivieren Sie diese Ansicht, indem Sie im Menü VIEW • TOOL WINDOWS • LOGCAT auswählen oder auf die entsprechende Schaltfläche am unteren Rand des AS-Fensters klicken. LOGCAT ist nichts anderes als das Protokoll aller Geschehnisse auf dem Gerät, und selbstverständlich gehören auch Fehlermeldungen dazu. Grundsätzlich besteht jede Meldung aus einer oder mehreren Zeilen, die alle mit Datum und Uhrzeit versehen sind (siehe Abbildung 3.11).

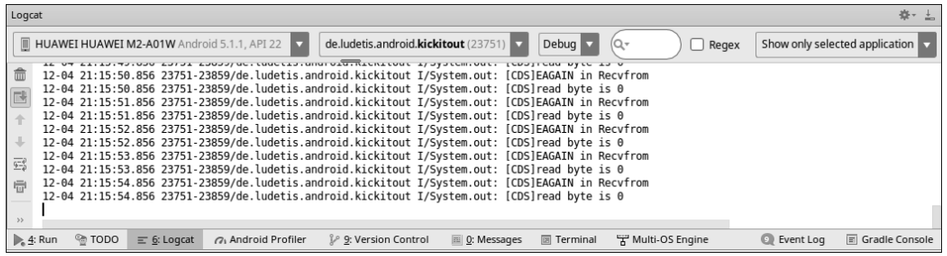

Abbildung 3.11 Die »Logcat«-View zeigt Ihnen, was auf dem Handy alles geschieht.

Außerdem verfügt jeder Eintrag über eine Protokollstufe (Log level). Es gibt fünf Stufen mit absteigender Wichtigkeit, die in unterschiedlichen Farben dargestellt werden:

- E = Error
- W = Warnung
- I = Info
- D = Debug
- V = Verbose (ausführlich)

Die Einträge verraten Ihnen außerdem ihren Ursprung über die Prozess-ID (PID) und ein TAG. Der Rest jeder Zeile ist für die eigentliche Meldung reserviert.

Leider ballert Android das Protokoll hauptsächlich mit Meldungen voll, die Sie gerade nicht besonders interessant finden. Aber das Logcat-Fenster bietet eine Reihe von Möglichkeiten, sich in dem ganzen Durcheinander zurechtzufinden.

Wählen Sie beispielsweise im oberen mittleren Pop-up Error. Dies beschränkt die Ausgaben im Fenster auf Zeilen mit ebenjener Protokollstufe *Error*. Falls Ihre App ein »Tut uns leid!« erzeugt, ist diese Einstellung eine gute Idee. Zum Vorschein kommt dann eine ganze Reihe von Zeilen, die zu dem Fehlerereignis gehören. Diese Zeilen nennt man *Stacktrace*.

Sie lesen den Stacktrace normalerweise von oben nach unten. Als Erstes steht da die eigentliche Fehlermeldung, die das »Tut uns leid!« verursacht hat. Sie lautet meistens `FATAL EXCEPTION: main`. Ursache ist immer eine Exception, die entweder im Android-System selbst oder innerhalb Ihrer App aufgetreten ist, aber mit keinem `catch`-Block gefangen wurde.

Die zweite Zeile des Stacktrace enthält die genaue Bezeichnung der aufgetretenen Exception, z. B. `ActivityNotFoundException`. Wenn Sie großes Glück haben, ist die Fehlersuche an dieser Stelle schon beendet: Manchmal können Sie aus dieser Meldung sofort auf die Ursache schließen. Im vorliegenden Beispiel wurde versucht, eine Activity zu starten, die nicht im Manifest angemeldet ist. Dies ist ein häufiger Fehler – so häufig, dass die Android-Macher sogar die Lösung mit in die ausführliche Meldung geschrieben haben: »Have you declared this activity in your AndroidManifest.xml?« Natürlich ist die Fehlersuche nicht immer so einfach.

Schauen Sie sich ein weiteres Beispiel an:

```
FATAL EXCEPTION: main
java.lang.RuntimeException: Unable to create service de.androidnewcomer.
weginsbuero.WegAufzeichnungsService: java.lang.NullPointerException
        at android.app.ActivityThread.handleCreateService(
           ActivityThread.java:2076)
        at android.app.ActivityThread.access$2500(
           ActivityThread.java:123)
        at android.app.ActivityThread$H.handleMessage(
           ActivityThread.java:993)
        at android.os.Handler.dispatchMessage(Handler.java:99)
        at android.os.Looper.loop(Looper.java:123)
        at android.app.ActivityThread.main(ActivityThread.java:3839)
        at java.lang.reflect.Method.invokeNative(Native Method)
        at java.lang.reflect.Method.invoke(Method.java:507)
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.
           run(ZygoteInit.java:841)
        at com.android.internal.os.ZygoteInit.main(
           ZygoteInit.java:599)
        at dalvik.system.NativeStart.main(Native Method)
Caused by: java.lang.NullPointerException
        at de.androidnewcomer.weginsbuero.WegAufzeichnungsService.
           onCreate(WegAufzeichnungsService.java:33)
        at android.app.ActivityThread.handleCreateService(
           ActivityThread.java:2066)
        ... 10 more
W/ActivityManager( 1695): Force finishing activity de.androidnewcomer.
weginsbuero/.WegInsBueroActivity
```

In diesem Fall ist offenbar das Starten des sogenannten `WegAufzeichnungsService` fehlgeschlagen, und zwar mit einer `NullPointerException`. Der Code versucht, ein Objekt zu verwenden, das `null` ist, also uninitialisiert. Jetzt müssen Sie den Rest des Stacktrace lesen, um der Ursache auf die Spur zu kommen.

Jede Zeile bezeichnet eine Stelle im Code, an der eine andere Methode aufgerufen wird, und zwar rückwärts. Sie sehen also zuerst die unterste Methode, dann die Stelle, an der diese Methode aufgerufen wurde, etc. Sie erkennen, dass sich das alles in Klassen abspielt, die Sie nicht geschrieben haben – bis es an einer Stelle heißt: `Caused by:`.

Android hat hier die `NullPointerException` gefangen und eine neue daraus gemacht. Hier, tief unten, finden Sie die ursprüngliche Exception und auch den Verweis auf Ihren eigenen Code. In diesem Beispiel war die fehlerhafte Zeile in *WegAufzeichnungsService.java*, Zeile 33. Dort steht:

```
weg.clear();
```

Die Ursache für eine `NullPointerException` ist fast immer, dass Sie auf eine Methode oder ein Attribut einer nicht initialisierten Variablen zugreifen. Das ist hier offensichtlich `weg`.

Suchen Sie die Stelle, an der `weg` deklariert wird. In diesem Beispiel ist es:

```
public static List<Location> weg;
```

Prüfen Sie anschließend, ob die Variable je mit irgendeinem Objekt initialisiert wurde, bevor der Zugriff in Zeile 33 erfolgt. Das ist nicht der Fall, also liegt der Fehler auf der Hand. Die Initialisierung fehlt, und richtig lautet die Deklaration:

```
public static List<Location> weg = new ArrayList<Location>();
```

Ich kann Ihnen hier natürlich nicht jede mögliche Exception erläutern, dazu gibt es zu viele. Nur so viel: Wenn Sie minutenlang ratlos auf Ihren Code starren, dann sollten Sie die Stelle einem Bekannten zeigen oder eine Kaffeepause einlegen und danach noch mal draufschauen. Irgendwann finden Sie den Fehler. Und verlassen können Sie sich auf eines: Mit den Java-Stacktraces kommen Sie Fehlern viel schneller auf die Spur als Generationen von Entwicklern vor Ihnen, die sich mit weniger gesprächigen Programmiersprachen herumschlagen mussten.

Logs von anderen Geräten

Es ist ziemlich unpraktisch, wenn Ihnen ein Benutzer einen Fehler meldet, den Sie auf Ihrem Gerät aber partout nicht nachvollziehen können. Tatsächlich ist nicht jedes Android-Gerät gleich; auf manchen mag eine App laufen, die auf anderen nur »Tut uns leid!« sagt. Für den Fall, dass Sie des Besitzers nicht habhaft werden können, um dessen Handy an Ihr Android Studio zu stöpseln, hilft Ihnen Google: Wenn Sie eine App bei *Google Play* veröffentlicht haben, können Benutzer Fehlerberichte einsenden. Das ist eine Möglichkeit, die nicht jeder Benutzer wahrnimmt, denn er kann die Frage des Handys, ob er einen Bericht senden möchte, natürlich ablehnen. Sie finden diese Berichte in Ihrer Entwicklerkonsole auf der Play-Webseite, die ich Ihnen in Abschnitt 12.2.3, »Fehlerberichte«, vorstellen werde. Die Berichte enthalten meist aussagekräftige Stacktraces und selbstverständlich auch die Versionsnummer der App, in der sie aufgetreten sind.

Da Sie auf nicht gerooteten Geräten die Logs von Fremd-Apps nicht mitlesen können, gibt es leider keine einfachere Möglichkeit.

3.6.2 Logging einbauen

Sie brauchen nicht unbedingt zu warten, bis ein fataler Fehler auftritt und Android einen Stacktrace ins Protokoll schreibt – Sie können auch ohne Weiteres selbst von Ihrer App aus Informationen ausgeben. Man nennt das *Logging*. Für kompliziertere Anwendungen ist es fast unerlässlich – vor allem wenn Berechnungen im Hintergrund ablaufen, sodass weder ein anderer Benutzer noch Sie selbst auf Anhieb sehen können, was genau geschieht.

Um Zeilen ins Protokoll zu schreiben, benutzen Sie statische Methoden der Klasse `Log`. Für jede Protokollstufe gibt es eine Methode. Fehlermeldungen können Sie beispielsweise wie folgt ausgeben:

```
Log.e("MyTag", "Fehlerbeschreibung");
```

Die Logging-Methoden erwarten zwei oder drei Parameter. Das genannte Beispiel ist eine einfache Variante mit zwei Parametern. Der erste ist das *Log-Tag*, das Sie bereits aus der LOGCAT-Ansicht kennen. Es dient dazu, zu erkennen, dass die Meldung von Ihrer App kommt. Wählen Sie einen eindeutigen Bezeichner, und definieren Sie ihn am besten als Konstante:

```
public static final String LOGTAG = "MeineApp";
```

Das vermeidet Tippfehler, die einzelne Einträge vermeintlich vor Ihnen verstecken.

Parameter Nummer zwei ist eine beliebige Ausgabe, die Ihnen bei der Untersuchung des Verhaltens Ihrer App hilft. Beispielsweise können Sie mit passenden Log-Kommandos sichtbar machen, wann eine Activity startet oder beendet wird:

```
public void onCreate() {
   super.onCreate();
   Log.d(LOGTAG, "Activity startet");
   ...
}
```

```
public void onDestroy() {
   Log.d(LOGTAG, "Activity beendet");
   super.onDestroy();
}
```

Natürlich können Sie auch aktuelle Zahlenwerte ausgeben, die Aufschluss über den Programmablauf geben:

```
public void onLocationChanged(Location location) {
   weg.add(location);
   Log.d(LOGTAG, "Aktuelle Weglänge: " + weg.size() );
}
```

Sie können mit der LOGCAT-View die Ausgabe auf Ihre eigene App begrenzen, indem Sie einen *Log-Filter* einrichten. Klicken Sie links im zweiten Pop-up Ihre App an, und wählen Sie rechts SHOW ONLY SELECTED APPLICATION aus.

Achten Sie darauf, dass Sie innerhalb der Log-Ausgaben keine komplizierten Berechnungen durchführen. Ich habe schon von Applikationen gehört, die zehnmal schneller wurden, wenn man das Logging ausschaltete.

Die Faustregel lautet: Protokollieren Sie während der Entwicklungsphase alles, was Sie brauchen, *aber nicht mehr*. Wenn Sie sicher sind, dass ein Programmteil korrekt arbeitet, werfen Sie alle Log-Befehle raus, oder machen Sie sie mithilfe einer `boolean`-Konstanten global abschaltbar:

```
if(LOGGING) Log.d(LOGTAG, "Nebensächliche Information");
```

Lassen Sie nur das Logging in der App, das Sie wirklich brauchen. Denken Sie außerdem daran, keine sicherheitskritischen Informationen zu loggen, beispielsweise Ihren API-Key von *Google Maps* oder gar eine Banking-PIN. Denn jeder Benutzer mit gerootetem Handy kann diese Informationen prinzipiell sehen. Verraten Sie im Protokoll nichts über Ihre App, was gegen Sie verwendet werden könnte.

3.6.3 Schritt für Schritt debuggen

Manchmal hilft auch kein Log weiter. Sie wissen überhaupt nicht, was Ihre App gerade tut? Sie würden gerne genau sehen, welcher Code gerade ausgeführt wird?

Auch das geht mit Android Studio. Überlegen Sie sich zunächst, an welcher Stelle der Ablauf der App unterbrochen werden soll. Klicken Sie dann im Editor links von der gewählten Zeile auf den Rand des Fensters. Es entsteht ein roter Breakpoint. An dieser Stelle wird die App stehen bleiben und Ihnen tiefe Einblicke gewähren. Allerdings klappt das nicht, wenn Sie die App wie bisher mit RUN starten. Wählen Sie stattdessen DEBUG oder das Icon mit dem grünen Käfer.

Sobald Ihre App die betreffende Stelle erreicht, pausiert sie, und AS schaltet die DEBUG-Ansicht ein (siehe Abbildung 3.12).

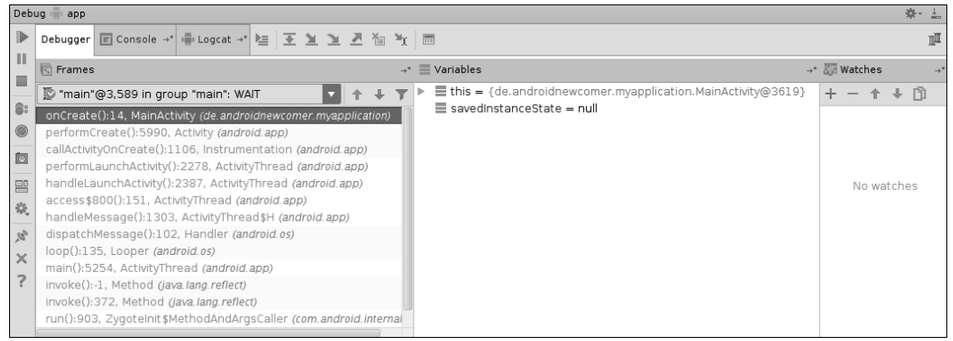

Abbildung 3.12 Wenn die App am Breakpoint stoppt, schaltet AS die »Debug«-Ansicht frei.

Die DEBUG-Ansicht erlaubt Ihnen genaue Einblicke in die App. Links sehen Sie den Stacktrace: Jede Zeile zeigt den Namen der Funktion und der Klasse, in der sich die App gerade befindet. Die zweite Zeile zeigt, von wo die Funktion in der ersten aufgerufen wurde – und so weiter.

In der Mitte sehen Sie die gerade verwendeten Variablen samt Inhalt. Diesen blendet AS sogar im Quellcodefenster ein, und zwar in Form von hellgrauen Texten am Ende jeder relevanten Zeile. Sie können sich in Ruhe die Werte ansehen – vielleicht ist einer anders, als Sie es erwartet haben. An dieser Stelle können Sie sogar Werte ändern, um zu untersuchen, wie Ihre App darauf reagiert (Rechtsklick und SET)!

Am linken Rand der DEBUG-View finden Sie mehrere Icons, die es erlauben, die App direkt zu steuern: Mit dem grünen Pfeil können Sie die App weiterlaufen lassen, bis sie auf den nächsten Breakpoint trifft. Oft sehr hilfreich sind die Icons mit den kleinen Pfeilen weiter rechts: Sie erlauben es, jeweils nur die nächste Zeile auszuführen. Auf diese Weise können Sie Schritt für Schritt beobachten, was Ihre App gerade tut, wie sich Variableninhalte ändern und welche Methoden Ihre App aufruft.

Breakpoints können Sie löschen, indem Sie erneut darauf klicken; alternativ finden Sie eine Liste in der BREAKPOINTS-View (viertes Icon von oben am linken Rand). Um das Debugging zu beenden, klicken Sie auf das Icon mit dem roten Quadrat.

Es kostet eine Menge Zeit, und außerdem gehört eine ordentliche Portion Geschick dazu, Breakpoints an der richtigen Stelle zu setzen, um mit dem schrittweisen Debugging Fehler zu identifizieren – aber wenn es hart auf hart kommt, haben Sie keine andere Möglichkeit.

Beachten Sie, dass Android Studio zwar normalerweise ziemlich stabil läuft, aber mit der Zeit intern eine ganze Menge Datenmüll anhäuft. Wenn Sie den Eindruck haben, dass das System holpert, liegt es möglicherweise nicht an Ihrer App. Wählen Sie den Menüpunkt File • Invalidate Caches/Restart, um den Besen zu schwingen und AS frei von Ballast neu zu starten. Das hilft erstaunlich oft, vor allem wenn Sie wenig freien RAM-Speicher im Rechner haben.

Auf einen Blick

Inhalt

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Uwe Post

Android-Apps entwickeln für Einsteiger

388 Seiten, broschiert, 7. Auflage, Januar 2018
24,90 Euro, ISBN 978-3-8362-6139-5

www.rheinwerk-verlag.de/4586

Uwe Post, Jahrgang 1968, überlebte in seiner Karriere als Entwickler und IT-Berater alle dreizehn Quellcode-Höllen. Vielleicht, weil er ein Diplom in Physik und Astronomie hat. Er hält Schulungen und Vorträge und schreibt neben IT-Büchern auch Science-Fiction-Romane – davon sogar deutlich mehr. Sein Roman »Walpar Tonnraffir und der Zeigefinger Gottes« gewann den Kurd-Laßwitz-Preis und den Deutschen Science-Fiction-Preis.