Fable Run

Eine Laufapp für Kinder

Hochschule für Angewandte Wissenschaften Hamburg

Department Medientechnik

Studiengang Media Systems

Projektausarbeitung Mobile Systeme

Betreut durch: Prof. Dr. Andreas Plaß

Sebastian Bohn Matr.Nr. 2036605

Johannes Bagge Matr.Nr. 2076759

Inhaltsverzeichnis

1. Planung

Die geplante, native Android-Application soll die Fortbewegungsgeschwindigkeit des Anwenders messen und mit der Geschwindigkeit verschieden schneller Tiere verglichen werden. Eine Geschwindigkeit von ca. 5 km/h soll z.B. zu einer Ausgabe à la “Du warst so schnell wie eine Schildkröte!” führen. Die App soll sich an Kinder ab dem lauffähigen Alter richten, aber auch ältere Kinder (Grundschule) ansprechen. Zur Geschwindigkeitsmessung wird das Global Positioning System, kurz GPS, verwendet.
Das Interface soll einfach und übersichtlich gehalten werden und sich auf die wesentlichen Informationen konzentrieren. Optimal wäre eine One-Button-Bedienung, die das Starten des Messzeitraums und das Beenden Selbigen erlaubt. Bei Beendigung eines gemessenen Laufs soll das Ergebnis mit einer Datenbank verschiedener Tiere und ihrer Geschwindigkeit verglichen und als Ergebnis ausgegeben werden. Während des Laufens soll ein Echtzeitvergleich zwischen der Geschwindigkeit des Anwenders und vorgegebenen Tieren stattfinden, um einen visuellen Anreiz zu bieten. Beispielsweise könnte der Nutzer gegen einen Hasen und eine Schildkröte starten und sich visuell stets zwischen den beiden befinden, wenn er langsamer als der Hase, aber schneller

als die Schildkröte ist (siehe Abb. 2.). Die App im Ganzen soll Kinder an das abstrakte Konstrukt einer Geschwindigkeitsmessung heranführen und ein Verhältnis ihrer eigenen Geschwindigkeit visualisieren. Hinzukommend soll es eine spielerische Möglichkeit bieten kompetitiv gegen die Tierwelt anzutreten und sich dabei selbst stets zu steigern.

Aus diesem Konzept kristallisierte sich entsprechend der Fabel „Die Schildkröte und der Hase“ der Projektname „Fable Run“.

Eine Steigerung, in Bezug auf die Nutzung über die Kindesentwicklung hinweg, wäre denkbar. Zum Beispiel könnte nicht nur gelaufen, sondern auch andere Fortbewegungsarten genutzt werden. So kann ein jüngeres Kind auf einem Lauflernrad gegen langsamere Tiere antreten und ältere mit dem Fahrrad gegen schnellere Tiere fahren. Eine optionale Erweiterung ist eine Highscore-Liste, bzw. eine Auflistung der bereits geschlagenen Tiere.

Abb. 1. Startbildschirm

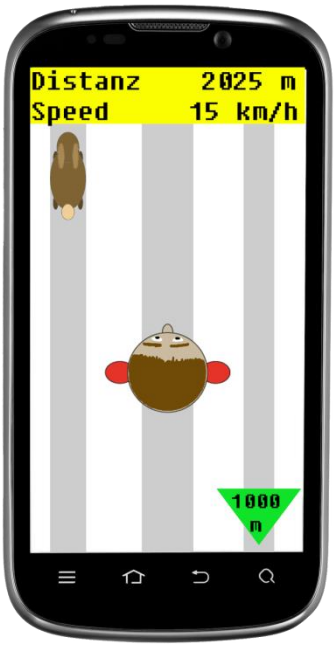

Abb. 2. Während der Messung

Abb. 3. Nach der Messung

2. Umsetzung

Insgesamt wurde sich bei der Umsetzung der App eng an die Planung gehalten. Die Oberfläche hat ein simples, übersichtliches Erscheinungsbild mit wenigen Bedienelementen und ein schlichtes schwarz-weißes Design.
Nach Berührung des ersten Bildschirms blenden die Bedienelemente der App ein und offenbaren das Nutzerinterface. Als zentrales Element wurde ein schwarzer Kreis auf weißem Grund gewählt, der zunächst das Logo und während des, bzw. nach dem Lauf das nächst langsamere Tier darstellt. Der Stil aller Bilder sind schwarze Silhouetten der entsprechenden Tiere, da sich so ein stimmiges Gesamtkonzept im Design ergibt. Das zentrale Bild dient zusätzlich an zwei Stellen als Bedienelement und zwar zu Beginn, um die restlichen Bedienelemente einzublenden und nach einem Lauf, um eine neue Runde einzuleiten. Zur Hilfestellung werden entsprechende Hinweise in Textform angezeigt. Ansonsten stehen dem Nutzer zwei, dem Design entsprechende Buttons am unteren Bildschirmrand zur Verfügung. Einer zum Starten oder Pausieren und einer um die Messung zu Stoppen.
Während der Entwicklungsphase wurden drei Activities entwickelt, eine StartActivity, die den Nutzer begrüßt, eine MainActivity, die für die Messung verantwortlich ist und eine ResultActivity, die das Ergebnis darstellt und einen Reset der App durchführt. Im Rahmen einer vollständigen Restrukturierung des Designs, von bunten Elementen zu genanntem schlichten schwarz-weiß, wurden zwei Activities verworfen und alles in einer einzigen Activity vereint, um dem Nutzer ein möglichst flüssiges Erlebnis zu bieten. Dazu werden überlagerte Elemente je nach Zustand der Anwendung ein- und ausgeblendet.

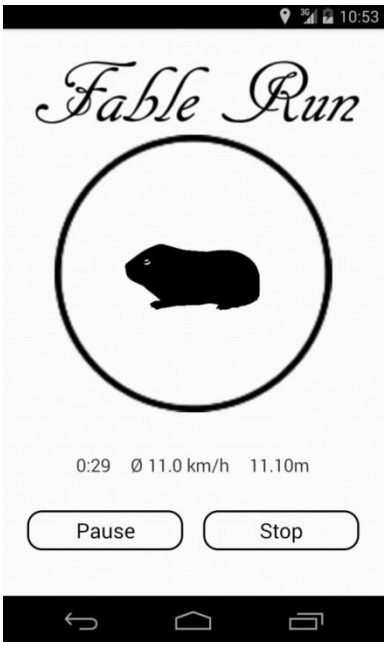

Abb. 4: Start-Ansicht

Abb. 5.: Während der Messung

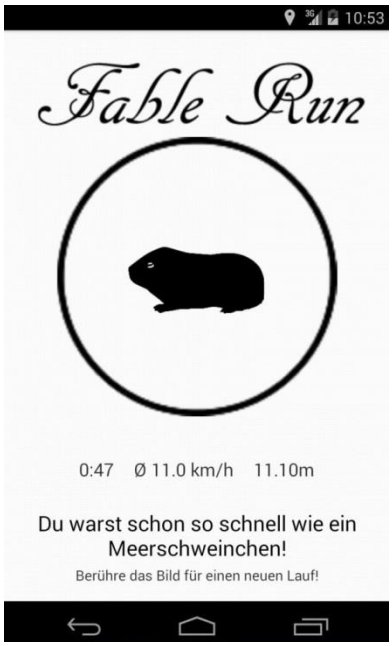

Abb. 6.: Ergebnis-Ansicht

3. Aufbau des Codes

Zur Messung der zurückgelegten Strecke verwendet die App aktuell ausschließlich GPS. Als zentrale Methode wird die *.onLocationChanged()*-Methode eines implementierten *LocationListener*-Objekts herangezogen. Die Methode wird innerhalb eines Zeitintervalls aufgerufen (aktuell einmal pro Sekunde), aber unabhängig einer wählbaren Distanz und misst in diesem Intervall, ob eine Änderung der GPS-Position stattfand. Innerhalb der Methode wird die Distanz zwischen dem aktuellen und dem Standort des vorangegangenen Intervalls gemessen und stetig aufaddiert. Dafür wird die Methode *Location.distanceTo()* genutzt, die einen Wert in Meter zurückgibt, unter Berücksichtigung des World Geodetic System 1984 (WGS84). Zusammen mit der vergangenen Zeit seit Laufstart, die ein parallel laufender Thread liefert, kann so die Durchschnittsgeschwindigkeit berechnet werden. Diese wird anschließend zusammen mit der vergangenen Zeit und zurückgelegten Strecke im Userinterface aktualisiert.
Mit der aktualisierten Durchschnittsgeschwindigkeit als Argument, liefert die aufgerufene Methode *.findSlowerAnimal()* ein Objekt der Klasse *Animal* zurück, welches Namen, Imagereferenz und Geschwindigkeit des Tieres beinhaltet, welches kleiner gleich der Geschwindigkeit des Nutzers ist. Dieses wird verwendet, um während des Laufs und in der Ergebnisansicht das Bild und den Namen des richtigen Tieres anzuzeigen.
Abgesehen von diesem zentralen Inhalt, wurden einige Abfragen eingebaut, die sicherstellen, dass der Nutzer vor dem Starten eines Laufes sowohl sein GPS-Modul aktiviert, als auch ein beständiges GPS-Signal hat. Da es keine Methode zu geben scheint, die direkt auf den Zustand des GPS-Moduls Zugriff hat (Signal fix oder nicht), wurde ein entsprechendes Workaround implementiert, das alle drei Sekunden überprüft, ob das GPS-Signal noch fixiert ist. Ein Lauf wird jedoch nicht abgebrochen, sobald das Signal verloren geht, sondern es wird mit der zuletzt gemessenen *Location* gearbeitet und auf die nächste Verbindung gewartet.
Da nur eine Activity für alle Ansichten verwendet wird, werden Ein-/Ausblende-Effekte für die Anzeigeelemente genutzt. Geplant war die Elemente jeweils nicht einfach verschwinden oder erscheinen zu lassen, sondern eine Fade-Animation zu verwenden. Zu diesem Zweck wurden dem Projekt zwei entsprechende .xml-Dateien hinzugefügt(FadeIn.xml / FadeOut.xml). Leider lässt sich diese Animation ausschließlich auf das BannerView-Objekt auswirken, welches den App-Namen darstellt. Andere Objekte vom gleichen Datentyp zeigten bisher keine Wirkung durch die Animation.

Alle angezeigten Texte in der App verwenden String-Elemente als Ressource, welche in der strings.xml-Datei definiert sind. Dieses Vorgehen entspricht Googles Vorgaben und ermöglicht außerdem eine leichte Lokalisation der Anwendung.
Neben der *MainActivity*-Klasse wurde eine *Animal*-Klasse erstellt, welche dazu dient Informationen zu jeweils einem Tier in der Datenbank zu speichern. Diese Klasse implementiert außerdem das *Comparable-Interface*, damit die Tiere einfach entsprechend ihrer Geschwindigkeit sortiert werden können. Die Datenbank wurde mit einer *ArrayList*<> realisiert, die entsprechende *Animal*-Objekte dynamisch speichert. Die angezeigten Bildressourcen sind als *drawables* im Projekt eingebunden.

4. Arbeitsweise

Zum gemeinsamen Arbeiten am Projekt wurde die freie Versionskontrolle Git, bzw. dessen Implementierung GitHub verwendet. Zunächst war das Repository unter *https://github.com/Johannes-HAWHH/FableRun.git* unter dem Profil von Johannes Bagge zu finden. Im Verlauf der Arbeiten wurde ein neues Repository erstellt, das unter dem Profil von Sebastian Bohn (*https://github.com/ SebastianBone/FableRun.git)* zu finden ist. Der Entschluss dazu war eine Notlösung, da es von Beginn an diverse Probleme im Umgang mit Git gab, obwohl sich, speziell um solche Probleme zu vermeiden, bereits vor Beginn der Arbeiten intensiv mit den Grundlagen Gits auseinandergesetzt wurde. Zur Betrachtung des Arbeitsfortschritts bzw. der Commit-History sollten beide Repositories herangezogen werden.
Der Code wurde komplett mit dem Entwicklungswerkzeug Eclipse geschrieben.
Die Bilder wurden mit Adobes Photoshop und Illustrator erstellt.
Wie erwähnt, hat das gemeinsame Arbeiten mit Git nicht so gut funktioniert wie geplant. Im Detail heißt das, dass sich durch eine Menge unterschiedlicher Fehler durchgearbeitet werden musste und teilweise deutlich mehr Zeit in die Behebung dieser Probleme gesteckt werden musste, die ursprünglich für das eigentliche Projekt gedacht war. Häufigster Fehler war eine Konflikt-Fehlermeldung, wenn gleichzeitig Änderungen an der gleichen Datei gemacht wurden. Eine einfache Methode diese Konflikte manuell zu lösen war nicht ersichtlich, so dass dann

mehrfach nichts anderes übrig blieb, als die Änderungen auf einem Rechner zu verwerfen und anschließend erneut einzufügen.

5. Probleme

Über die Probleme mit GitHub hinaus, warteten noch weitere Komplikationen, die sehr viel Zeit in Anspruch genommen haben. Ein kleiner aber schwerwiegender Bug, der fast von Beginn an die App jedes Mal nach dem Start sofort zum Absturz brachte, bremste den ersichtlichen Fortschritt aus. Da der Fehler nicht sofort auffindbar war, jedoch das Projekt vorankommen sollte, wurde anschließend ein Großteil der App “blind” (ohne Tests) entwickelt. Als der Fehler später behoben werden konnte, kam folglich eine angesammelte Menge neuer Probleme. Leider war es bis zur Abgabe nicht möglich alle diese Probleme zu lösen, so ist die Berechnung der Durchschnittsgeschwindigkeit bis dato fehlerhaft. Dies führt dazu, dass keine korrekte Durchschnittsgeschwindigkeit errechnet werden kann, was wiederum zur Folge hat, dass die Anzeige des Tieres ins Leere läuft. Durch Debuggen konnte der Fehler jedoch (vermutlich) auf die Umrechnung der vergangenen Zeit von Millisekunden in Stunden eingegrenzt werden, da die Variable für die Stunden stets 0.0 enthält, was in der Berechnung der km/h natürlich zu einem Teilen durch 0 und zu NaN (Not a Number) führt.
Darüber hinaus konnten auch die Animationen, mit Ausnahme der BannerView, noch nicht realisiert werden. So wirkt das Anzeigen des Interfaces etwas stockhaft und erreicht noch nicht das runde und flüssige Nutzererlebnis, das angestrebt ist.

6. Ausblick

In Zukunft wird die App weiterentwickelt und die aufgezeigten Fehler versucht auszumerzen. Geplant ist weiterhin eine Benachrichtigung in der Statusleiste des Betriebssystems, wenn der

Nutzer die App “minimiert”, solange eine Messung läuft. In der Benachrichtigung können die relevanten Daten, sprich Geschwindigkeit, Strecke und Zeit, angezeigt werden, während ein Klick dergleichen zurück zur App führt.
Auch eine Erweiterung der Datenbank im oberen Bereich ist denkbar, da das schnellste Tier in der Datenbank momentan ein Bär mit 40 km/h ist. Für einen wirklichen Lauf oder auf dem Fahrrad reicht das natürlich, nicht jedoch, wenn die App in einem PKW, Zug, etc. verwendet werden soll.
Zudem erfuhren wir bei dem Präsentationstermin der Apps, im Austausch mit Kommilitonen die ähnliche Apps erstellt haben, von der Existenz der “Google Play Services”, die zur Positionsbestimmung, Berechnung der Geschwindigkeit und Überwachung des GPS-Signals verwendet werden. Dieses ermöglicht zusätzlich die Bestimmung der Position in geschlossenen Räumen, welche ein W-LAN Signal liefern. Nach einer Recherche über diesen Service wurde festgestellt, dass eine Menge der Probleme mit Hilfe dieses Dienstes gar nicht erst aufgekommen wären. Da innerhalb der Recherche meist nach expliziten Lösungen und Snippets gesucht wurde, entfiel diese Variante der Aufmerksamkeit.
Um diesen Service zu implementieren, muss nahezu die komplette Struktur der Anwendung neu geschrieben und angepasst werden. Für die Zukunft wird dies aber zu einer besseren Lösung führen, da hiermit genauere und stabilere Daten geliefert werden .

7. Abbildungsverzeichnis