Spielend Fremdwörter büffeln mit DashInEnglish

Moderne Smartphones und Tablets haben auf die heutige Jugend eine sehr grosse Anziehungskraft, die für spielerisches Lernen genutzt werden kann. Anstatt englische Vokabeln mithilfe von Karteikärtchen oder Wörterbüchern zu büffeln, lässt sich das Pflichtvokabular auf einem mobilen Gerät spielerisch erwerben. Die Android-App DashInEnglish basiert auf dem unkonventionellen Dasher-Schreibsystem, welches zur Texteingabe auf einem berührungsempfindlichen Bildschirm dient. Dabei fährt die Benutzerin mit einem Finger mit einer sanften Bewegung von einem Buchstaben zum nächsten und erhält dabei laufend Rückmeldung über die Korrektheit der angesteuerten Buchstaben.

Zdena Koukolíková, Carlo U. Nicola | carlo.nicola@fhnw.ch

Dasher wurde ursprünglich in der Inference-Gruppe von David MacKay an der Universität Cambridge entwickelt [WBM00] und von uns für die Android-Plattform in einer veränderten Form in Java von der Pike auf neu geschrieben [GKN12a/b]. Das Programm stellt ein Texteingabe-Interface zur Verfügung, das mit natürlichen, kontinuierlichen Zeigegesten geführt wird. Das Schreiben fühlt sich an, als ob man mit dem Finger einen virtuellen Wagen auf dem Bildschirm steuern würde; die Benutzerin fährt stets in die Richtung des Buchstabens, den sie ihrem Text als nächsten hinzufügen möchte (siehe Abb. 2). Dieses spielerische Schreiben dürfte Kinder und Jugendliche anregen, das wenig geliebte, aber durchaus notwendige Fremdwörterbüffeln mit mehr Spass in Angriff zu nehmen.

Im vorliegenden Beitrag stellen wir unsere Android-App *DashInEnglish* vor, welche die Grundzüge einer spielerischen Lernhilfe für Englischvokabeln illustriert. Diese App ist sowohl für SmartPhones als auch Tablets konzipiert. Um beide Endgerättypen gleichermassen gut zu unterstützen, sind ein paar Vorkehrungen bei der Implementierung zu treffen, die wir in diesem Artikel genauer vorstellen werden.

Aufbau des GUIs

Auf einem Nexus 7 Tablet sieht die grafische Benutzungsoberfläche von *DashInEnglish* wie in Abbildung 1 (Anfangsposition) oder in Abbildung 2 aus. Im linken Teil der Anwendung lässt sich aus einer Liste der verschiedenen Lektionen die gewünschte auswählen, die dann rechts auf dem Bildschirm erscheint. Im oberen Textfeld wird jeweils das zu übersetzende deutsche Wort angezeigt und im darunterliegenden Textfeld werden die bis dahin geschriebenen Buchstaben der englischen Übersetzung dargestellt.

In Abbildung 2 ist die erste Lektion ausgewählt worden und die Benutzerin wird aufgefordert, eine englische Übersetzung („letterbox“) für das Wort „Briefkasten“ mit Hilfe des auf der rechten Seite angezeigten Dashers einzugeben. Zum aktuellen Zeitpunkt hat sie bereits den ersten richtigen ‚l‘ Buchstaben ausgewählt und der *Prediction Partial Matching* (PPM) Algorithmus die drei wahrscheinlichsten Nachfolgebuchstaben ‚a‘, ‚e‘ und ‚o‘ vorausberechnet. Quadrate unterschiedlicher Farben stellen verschiedene Buchstaben dar. Nun muss die Benutzerin mit ihrem Finger das Quadrat des nächsten Buchstabens in Richtung des Zentralkreuzes (siehe Abb. 1) lenken. Sobald das Zentralkreuz in einem Buchstabenquadrat liegt,

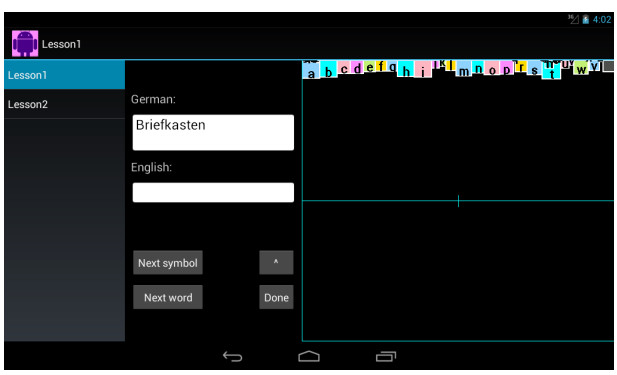

Abbildung 1: Dasher-Anfangsposition für ein neues Wort

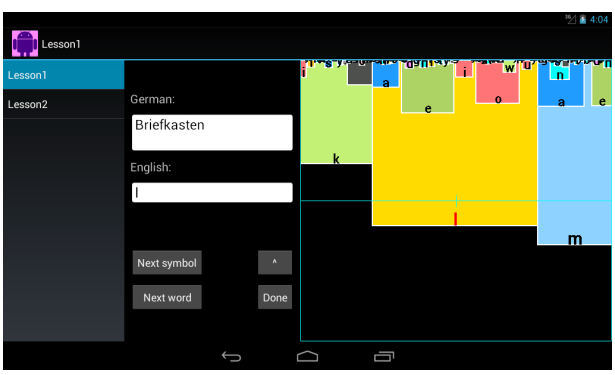

Abbildung 2: Eingabefläche von DashInEnglish auf einem Android Nexus 7 Tablet in horizontaler Lage

wird der entsprechende Buchstabe automatisch ins untere Textfeld übernommen.

Die vier Eingabeknöpfe unterhalb des Textfelds haben die folgenden Bedeutungen:

- „Next symbol“: bewegt Dasher zum nächsten richtigen Buchstaben;
- „Next word“: wählt zufällig ein neues Wort aus dem Vokabular der aktuellen Lektion;
- „^“ stellt von Klein- auf Grossbuchstaben um und umgekehrt;
- „Done“: verlässt die App.

Texteingabe

Wie zuvor erwähnt, erfolgt die Texteingabe des englischen Textes im rechten Teil der App. Mit einer Fingerberührung auf die obere Hälfte der Oberfläche beginnt Dasher sich in Richtung des Fingers vorwärts zu bewegen, wobei sich die angepeilten Buchstabenquadrate laufend vergrössern. Sobald ein Quadrat das Zentralkreuz bedeckt, wird der entsprechende Buchstabe in das Englischtextfeld übernommen. Wenn der Buchstabe korrekt ist, wird er innerhalb des Dashers farblich hervorgehoben. Um das positive Feedback zu erhöhen, ertönt gleichzeitig ein akustisches Feedback. Wenn der Buchstabe dagegen nicht richtig ist, kann zwar weitergeschrieben werden, die Buchstaben bleiben aber schwarz.

Falls man einen Tipp braucht oder sich total „verlaufen“ hat, kann man jederzeit den Knopf „Next symbol“ drücken. Dadurch erscheint der auf den schon geschriebenen Text nächstfolgende, korrekte Buchstabe sowohl in der Texteingabe als auch im Dasher. Von da an kann man wieder mittels Fingerbewegungen weiterschreiben oder wenn nötig den nächsten Buchstaben verlangen.

Wenn man einen falschen Buchstaben eingegeben hat, ist dies nicht weiter schlimm, da man sehr einfach einen anderen Pfad einschlagen kann. In der in Abbildung 2 dargestellten Situation könnte man z. B. den Dasher einfach in Richtung des Symbols ‚k‘ lenken und damit das bereits geschriebene ‚l‘ überschreiben. Dies geschieht sobald das ‚l‘-Quadrat das Zentralkreuz verlässt und das ‚k‘-Quadrat das Kreuz bedeckt. Wenn mehrere Buchstaben falsch sind, berührt man am besten die untere Hälfte der Dasher-Oberfläche und fährt den Dasher somit zurück. Sobald ein Quadrat das Zentralkreuz verlassen hat, wird der entsprechende Buchstabe gelöscht.

Das Navigieren im Dasher-Raum wird durch zusätzliche Gesten erleichtert. So lassen sich die Quadrate entweder nach links oder nach rechts bewegen, indem man rechts bzw. links auf der Mittellinie des Dashers drückt. Mit zwei Fingern ausgeführte Swipe-Bewegungen lassen den Dasher entweder ein- oder auszoomen. Nach einer sehr kurzen Einarbeitungszeit beherrscht man diese verschiedenen Optionen und das Navigieren auf dem Bildschirm wunderbar.

Die Eingabe eines Wortes wird als beendet betrachtet, wenn das drauffolgende Leerzeichen-Quadrat (ohne Symbol) das Zentralkreuz bedeckt. Sobald dies eintritt, ertönt das eingegebene Wort auf Englisch. Hierzu verwendet *DashInEnglish* die vom Android-System bereitgestellte, synthetische Sprache. Stattdessen liessen sich aber auch Aufnahmen einer menschlichen Stimme verwenden. Durch Betätigen des Knopfes „Next word“ kann das nächste Wort in Angriff genommen werden. Die Auswahl der Wörter geschieht zufällig, damit die Wörter unabhängig voneinander gelernt werden. Es wird jedoch garantiert, dass innerhalb einer Lektion alle gespeicherten Wörter behandelt werden.

Textvoraussage

Um das Schreiben zu vereinfachen und somit zu beschleunigen, benutzt Dasher ein prädiktives Sprachmodell der jeweiligen Sprache, das laufend den nächstfolgenden Buchstaben im gegebenen Textkontext vorschlägt. Man beachte, dass die Grösse der verschiedenen Quadrate, die den potenziellen nächstfolgenden Buchstaben darstellen, proportional zu der Wahrscheinlichkeit ist, mit welcher die entsprechenden Zeichen in der jeweiligen Sprache vorkommen.

Ward, Blackwell und MacKay [WBM00] haben die arithmetische Kodierung- und die statistische Modellierungsmethode, die ursprünglich zur Textkomprimierung entwickelt wurden, sehr elegant an die Berechnung der Wahrscheinlichkeit des nächsten im Text vorkommenden Buchstabens angepasst [AK]. Das hierfür benutzte Sprachmodell basiert auf dem *Prediction Partial Matching* (PPM) Algorithmus [CW84]. In diesem werden die nächsten Buchstaben anhand des vorangehenden Kontextes, d. h. der n vorherigen Buchstaben, vorgeschlagen. So ist in unserem Beispiel in Abbildung 2 die Wahrscheinlichkeit sehr gross, dass das nächste Zeichen entweder ein ’a’, ‘e’ oder ein ‘o’ sein wird, wenn man auf Englisch bereits das ‚l‘ geschrieben hat. Wir verwenden die vier letzten Buchstaben für die Vorhersage. Für Details zum angewendeten Sprachmodell verweisen wir auf [GKN12a/b].

Es ist offensichtlich, dass bei diesem Eingabesystem die ersten Buchstaben des gesuchten Wortes oft schwerer zu „erraten“ sind, als die letzten. Dies ist darauf zurückzuführen, dass die Wahrscheinlichkeitsberechnung für die ersten Buchstaben auf einem sehr kurzen Kontext beruht. Somit muss die Benutzerin anfangs gut überlegen, in welche Richtung sie Dasher navigieren soll. Schliesst das Wort hingegen mit einer in der jeweiligen Sprache gängigen Endung, so schlägt Dasher mit grosser Wahrscheinlichkeit die letzten Buchstaben von alleine korrekt vor. Aus Erfahrung kann man aber annehmen, dass zum Erlernen neuer Fremdwörter meistens die

ersten Buchstaben des jeweiligen Wortes wichtig sind; die letzten fallen einem dann oft wie von selbst ein.

Anpassung an die Bildschirmgrösse

Die Herausforderung dieses Projekts besteht darin, die als Android-Texteingabe-Service von uns bereits implementierte Dasher-Anwendung so zu adaptieren, dass *DashInEnglish* perfekt sowohl an Smartphones, als auch an die immer beliebteren Tablets angepasst ist.

Wir haben die Entwicklung mit Hilfe der neuesten Android Version 4.1 (API 16) durchgeführt [APIa]. Unsere App kann auf eine sehr einfache Art verändert werden, damit sie auch auf älteren Android-Versionen funktionsfähig ist [APIb].

Da die Bildschirme der SmartPhones und der Tablets unterschiedliche Grössen und Formfaktoren aufweisen, muss die Darstellung der Applikation an das jeweilige Gerät angepasst werden. Diese Flexibilität wird durch die in der Version 3.x eingeführten *Fragments* ermöglicht. Ein *Fragment* repräsentiert normalerweise einen wiederverwendbaren Teil eines *Activity User Interfaces*.

Wie aus den Abbildungen 1 und 2 ersichtlich ist, haben wir das Tablet-UI in zwei Fragmente aufgeteilt: das linke Fragment *(LessonsListFragment)* zeigt die Liste der verschiedenen Lektionen, das rechte *(DasherFragment)* zeigt das zu übersetzende deutsche Wort der entsprechenden Lektion und die Oberfläche der graphischen Text-Eingabe. Beide Klassen sind von *Fragment* abgeleitet. Je nach Bildschirmgrösse sind die zwei Fragmente entweder gleichzeitig oder aber nacheinander zu sehen. So wird auf einem Smartphone nur die Lektionsliste als Teil der Hauptaktivität *(DashInEnglishActivity)* gezeigt, das zweite Fragment erscheint aber erst nach Auswahl einer Lektion als Teil der neu aufgerufenen *DasherFragmentActivity* auf dem Bildschirm.

Da das Android-System erlaubt, nicht nur eigene Layouts für verschiedene Orientierungen, sondern auch für verschiedene Gerätgrössen zu entwerfen, wird der Entscheid, nur ein oder beide Fragmente gleichzeitig auf dem Bildschirm darzustellen, mit Hilfe der XML-Dateien *fragments_layout.xml* gefällt, die sich in den Layout-Resources Ordnern „res/layout-sw320dp“ (Listing 1) und „res/layout-sw600dp“ (Listing 2) befinden. Für kleine Bildschirme wird im ersten Layout nur das Liste-Fragment definiert, während für grosse Bildschirme beide Fragmente festgelegt werden.

In beiden Fällen wird automatisch das *LessonsListFragment* erstellt, wenn die Hauptaktivität *DashInEnglishActivity* ihr Layout in der Methode *onCreate()* mit Hilfe von *setContentView(R.layout.fragments_layout)* sozusagen aufbläht.

Nur im zweiten Fall aber wird das *FrameLayout* Dasher erstellt, das den Platz auf dem Bildschirm definiert und reserviert, worin das *DasherFragment* dargestellt wird. Wenn eine Lektion aus der Liste ausgewählt wird, wird im nachfolgenden Code, ermittelt, ob im aktuellen Layout

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <fragment
    class="ch.fhnw.android.dasher.english.LessonsListFragment"
    android:id="@+id/lessons"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
  />
</FrameLayout>
```

Listing 1: Layout für Smartphones

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <fragment
    class="ch.fhnw.android.dasher.english.LessonsListFragment"
    android:id="@+id/lessons"
    android:layout_weight="1"
    android:layout_width="0dip"
    android:layout_height="match_parent"
  />

  <FrameLayout android:id="@+id/dasher"
    android:layout_weight="4"
    android:layout_width="0dip"
    android:layout_height="match_parent"
  />
</LinearLayout>
```

Listing 2: Layout für Tablets

der Rahmen existiert, worin das *DasherFragment* dargestellt werden kann.

```
View dFrame = findViewById(R.id.dasher);
mDualPane = dFrame != null &&
dFrame.getVisibility() == View.VISIBLE;
```

Wenn kein *dFrame* existiert, wird wie gewohnt eine neue Aktivität *(DasherFragmentActivity)* aufgerufen, die die entsprechende Lektion samt der Dasher-Texteingabefläche darstellt. Auf einem grossen Bildschirm dagegen wird die Lektion rechts in Form eines Fragments abgebildet. In beiden Fällen verinnerlicht das *DasherFragment* die ganze Lerntätigkeit. Gleichzeitig wird der Name der gegenwärtigen Lektion in der oberen Leiste *(ActionBar)* angezeigt. Im Fall eines Tablets wird beim Anwählen einer neuen Lektion das aktuelle Fragment auf einem Stack (dem sogenannten *backstack*) zwischengelagert, damit es wiederhergestellt werden kann, wenn die Benutzerin auf die Back-Taste drückt.

Damit die App auf dem Tablet-Bildschirm immer korrekt dargestellt wird, auch dann wenn das Gerät während der Arbeit gedreht wird, muss der aktuelle Zustand in allen drei Hauptkomponenten in deren *onSaveInstanceState()* Methode gesichert werden. Dies ist notwendig, weil alle drei Komponenten beim Drehen zerstört werden. Der gegenwärtige Titel, d.h. der Name der dargestellten Lektion, wird im Bundle *outState* der Hauptaktivität *DashInEnglishActivity* gespeichert, und in *onCreate()* wieder in die *ActionBar* eingesetzt. Die beiden Fragmente müssen jeweils die Daten sichern, anhand derer ihr Zustand rekonstruiert werden kann. Im *LessonsListFragment* wird der Index der zuletzt ausgewählten Lektion in der Liste im Bundle abgelegt und in der Methode *onActivityCreated()* wieder als ausgewählt markiert, und die Hauptaktivität davon benachrichtigt. Das *DasherFragment* benötigt nicht nur die Eingaben aus den beiden Textfeldern, d.h. das aktuelle deutsche Wort und die Buchstaben, die bis dahin ins englische Textfeld geschrieben wurden, sondern auch zusätzliche Informationen,wie Name der Lektion, ihr Index in der Lektionsliste, Anzahl und Liste der bis dahin bearbeiteten deutschen Wörter und das korrekte englische Wort. Mit den gespeicherten Angaben wird das jeweilige Fragment in den Methoden *onCreate()* und *onActivityCreated()* von neuem erzeugt und rekonstruiert.

Auswählen einer Lektion

Der Lebenszyklus des *DasherFragments* muss auch darum sorgfältig behandelt werden, da wir die entsprechenden Fragmente beim Wechseln von einer Lektion zur anderen (in Form einer *FragmentTransaction*) auf den *Backstack* schieben, und diese mit Hilfe des Back-Tastendrucks wieder zum Leben erwecken wollen. Die so gestapelten Fragmente werden nicht völlig zerstört, sondern nur gestoppt: die zugehörige *View* wird aber dabei zerstört, da die Fragmente ja nicht mehr sichtbar sind. Deshalb müssen wir die Angaben, die zur Rekonstruktion der jeweiligen *View* notwendig sind, in der *onStop()* Methode des *DasherFragments* speichern. Es handelt sich hier lediglich um das zuletzt bearbeitete deutsche Wort und den bis dahin geschriebenen englischen Text, die als Instanzvariablen *mLastGermanWord* und *mLastEnglishWord* gesichert werden. Die Werte dieser Variablen werden dann in der Methode *onActivityCreated()* in die entsprechenden, neu erstellten Textfelder eingesetzt. Anhand dieser Angaben wird genau wie im oberen Fall der letzte Zustand der graphischen Texteingabefläche und der Dasher mit Hilfe der Methode *setDasherViewToInputText()* rekonstruiert, indem Dasher sozusagen bis zu dem zuletzt geschriebenem Buchstaben vorgerückt wird. Die restlichen Angaben sind in dem jeweiligen Fragment in Form weiterer Instanzvariablen gesichert.

Damit die beiden Fragmente möglichst unabhängig voneinander sind, wird die Kommunikation via die Hauptaktivität bewerkstelligt. Da das *LessonsListFragment* beim Auswählen einer Lektion die Hauptaktivität benachrichtigen muss, damit sie im *DasherFragment* dargestellt wird, definieren wir im *LessonsListFragment* das Callback-Interface OnLessonSelectedListener mit der einen Methode *onLessonSelected(String lesson, int index)*, welche dann in der *DashInEnglishActivity* implementiert wird.

In der Methode *onAttach()*, die aufgerufen wird, sobald das Fragment mit seiner Host-Aktivität verbunden worden ist, erhält *LessonsListFragment* eine Referenz auf eine Instanz der Interface-Implementierung (Listing 3).

```
@Override
public void onAttach(Activity activity) {
  super.onAttach(activity);
  try {
    mCallback = (OnLessonSelectedListener)activity;
  } catch (ClassCastException e) {
    throw new ClassCastException(activity.toString()
        + " must implement OnLessonSelectedListener");
  }
}
```

Listing 3: Registrierung der Hauptaktivität als Selection-Listener im LessonsListFragment

```
@Override
public void onListItemClick(ListView l, View v, int pos, long id) {
  String lesson = (String)l.getItemAtPosition(pos);
  // Hauptaktivität benachrichtigen: ein Element der Liste ausgewählt.
  mCallback.onLessonSelected(lesson, pos);
  mCurCheckPosition = pos;
}
```

Listing 4: Benachrichtigung der Hauptaktivität beim Auswählen einer Lektion

Beim Berühren eines Elements in der Lektionsliste wird somit die Aktivität mittels der Methode *onLessonSelected()* benachrichtigt, wie in Listing 4 gezeigt.

Die Hauptaktivität reagiert auf dieses Ereignis (siehe Listing 5), indem sie zuerst bestimmt, ob die Applikation auf einem Gerät mit grossem oder kleinem Bildschirm läuft, d.h. ob im Layout Platz für zwei oder nur ein Fragment reserviert ist. Im ersten Fall wird das *DasherFragment* auf dem Bildschirm dargestellt, sofern es nicht schon abgebildet ist, und der entsprechende Titel erscheint in der *ActionBar*-Leiste. Gleichzeitig wird die ganze Transaktion, d.h. hier das Ersetzen der Instanz des gegenwärtigen Fragments mit einer Instanz des neuen Fragments, auf dem *backstack* gespeichert, damit die vorherigen Fragmente später via Back-Taste erreichbar sind. Im zweiten Fall wird mittels eines *Intent* eine neue *DasherFragmentActivity* aufgerufen, die das entsprechende *DasherFragment* abbildet. Im *Intent* schicken wir die nötige Information, damit die richtige Lektion gezeigt wird.

Wenn die Benutzerin auf dem Nexus 7 Tablet die Back-Taste drückt, müssen die beiden Fragmente synchronisiert gezeigt werden, d.h. der Name der in der Liste ausgewählten Lektion muss mit dem Titel und mit der im *DasherFragment* tatsächlich abgebildeten Lektion übereinstimmen. Zu diesem Zweck haben wir uns des Interfaces *OnBackStackChangedListener* bedient, das in der Android Klasse *FragmentManager* zur Verfügung steht, und haben die abstrakte *onBackStackChanged()* Methode gemäss Listing 6 implementiert. Diese Methode wird aufgerufen, sobald der Inhalt des backstacks sich ändert und ist darum ideal, um den Titel in der *ActionBar*-Leiste zu aktualisieren und, wie in unserem Fall, die zwei Fragmente in Einklang zu bringen.

```
@Override
public void onLessonSelected(String lesson, int index) {
// Existiert ein Rahmen in der UI wohin das Dasher Fragment
// eingesetzt werden kann?
  View dView = findViewById(R.id.dasher);
  mDualPane = (dView != null && dView.getVisibility() == View.VISIBLE);
  if (mDualPane) {
    // Prüfen welches Fragment momentan gezeigt wird; ersetzen, wenn nötig.
    DasherFragment dF = (DasherFragment)
          getFragmentManager().findFragmentById(R.id.dasher);
    if (dF == null || !dF.getLessonName().equalsIgnoreCase(lesson)) {
      // Neues Fragment erstellen, das die Lektion zeigt.
      dF = DasherFragment.newInstance(lesson, dir.getAbsolutePath(), index);
      // Eine Transaktion ausführen: jegliches beliebige Fragment mit dem
      // neuen ersetzen.
      FragmentTransaction ft = getFragmentManager().beginTransaction();
      ft.replace(R.id.dasher, dF);
      // Das alte auf backstack schieben.
      ft.addToBackStack(null);
      ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
      ft.commit();
      // Titel setzen.
      setTitle(lesson);
      currTitle.setLength(0);
      currTitle.append(lesson);
    }
  } else {
    // Ansonsten neue Aktivität aufrufen, die die ausgewählte Lektion
    // in ihrem Dasher Fragment darstellt.
    Intent dI = new Intent(this, DasherFragmentActivity.class);
    dI.putExtra("lessonName", lesson);
    dI.putExtra("dir", dir.getAbsolutePath());
    dI.putExtra("lessonIndex", index);
    startActivity(dI);
  }
}
```

Listing 5: Reaktion der Hauptaktivität beim Auswählen einer Lektion

```
  getFragmentManager().addOnBackStackChangedListener(
     new FragmentManager.OnBackStackChangedListener() {
    @Override
    public void onBackStackChanged() {
      // Prüfen, welches Fragment momentan gezeigt wird.
      DasherFragment dF = (DasherFragment)
        getFragmentManager().findFragmentById(R.id.dasher);
      if (dF != null && dF.isVisible()) {
        setTitle(dF.getLessonName());
        currTitle.setLength(0);
        currTitle.append(dF.getLessonName());
        // Das List-Fragment benachrichtigen.
        LessonsListFragment lF = (LessonsListFragment)
          getFragmentManager().findFragmentById(R.id.lessons);

        if (lF.mCurCheckPosition != dF.getLessonIndex()) {
          // Das Element als ausgewählt markieren.
          lF.getListView().setItemChecked(dF.getLessonIndex(), true);
          lF.mCurCheckPosition = dF.getLessonIndex();
        }
      } else {
        // Alle Fragmente sind geschlossen.
        setTitle(appTitle);
        currTitle.setLength(0);
        currTitle.append(appTitle);
        // Das List-Fragment benachrichtigen.
        LessonsListFragment lF = (LessonsListFragment)
          getFragmentManager().findFragmentById(R.id.lessons);
        // Kein Element ausgewählt.
        lF.getListView().setItemChecked(-1, true);
        lF.mCurCheckPosition = -1;
      }
    }
  });
}
```

Listing 6: Synchronisation der beiden Fragmente und Aktualsierung des Titels

Zusammenfassung

Basierend auf einem sehr intuitiven und einfachen Texteingabe-Interface haben wir als Demonstration dessen Anwendungsmöglichkeiten eine Applikation entwickelt, die als Hilfsmittel zum Erlernen von Fremdsprachewörtern dienen und für jede beliebige Sprache angepasst werden könnte. Wir sind überzeugt, dass der Einsatz neuer Technologien die Motivation von Kindern und Jugendlichen erheblich steigern könnte, unbeliebte, aber doch sehr notwendige Aufgaben spielend zu bewältigen.

Referenzen

[AK] Arithmetische Kodierung und PPM: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/.

[APIa] Fragments: http://developer.android.com/guide/components/fragments.html

[APIb] Building a Dynamic UI with Fragments: http://developer.android.com/training/basics/fragments/index.html

[CW84] Cleary, J.G., Witten, I.H. Data Compression Using Adaptive Coding and Partial String Matching, IEEE Trans. on Comm., Vol. COM-32, No. 4, 1984.

[GKN12a] Gasser, M., Koukolíková, Z., Nicola, C.U. Texteingabe als Spiel, Android User, 01, 106-110, 2012.

[GKN12b] Gasser, M., Koukolíková, Z., Nicola, C.U. Texteingabe als Spiel – Teil 2, Android User, 03, 108-113, 2012.

[WBM00] Ward, D.J., Blackwell, A.F., MacKay, D.J.C. Dasher—a data entry interface using continuous gestures and language models, UIST '00 Proc. of the 13th annual ACM symposium on User interface software and technology, 2000.