MMI 2: Mobile Human-Computer Interaction

Übung 4

Prof. Dr. Michael Rohs

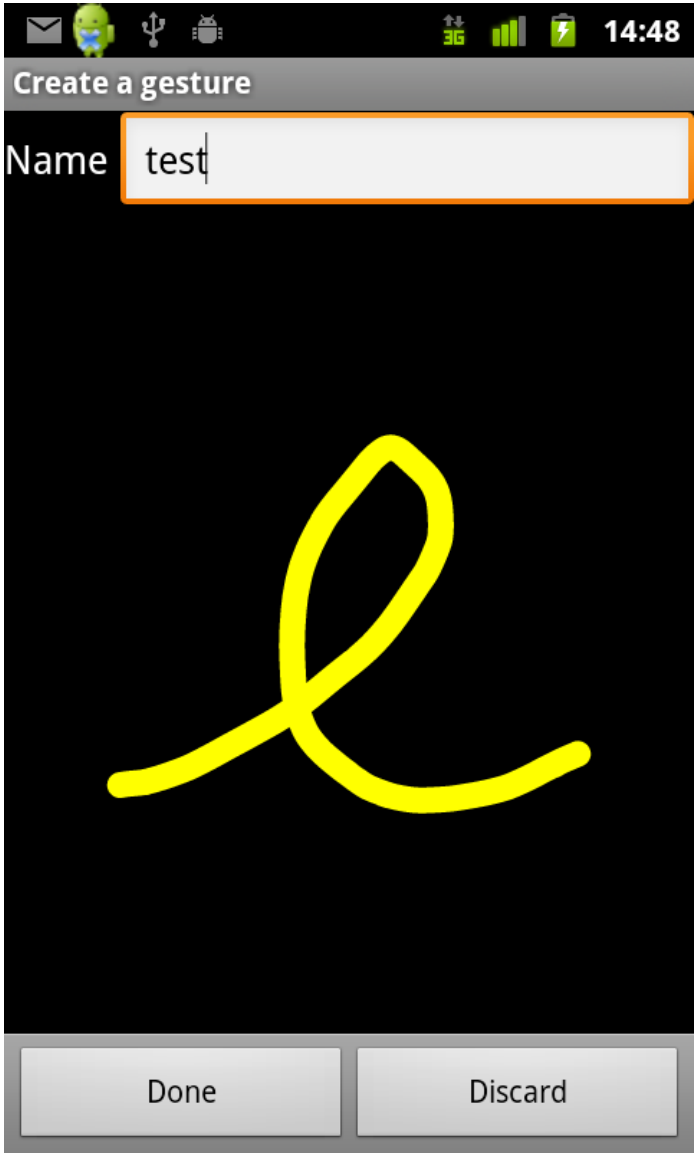
michael.rohs@ifi.lmu.de

Mobile Interaction Lab, LMU München

Gestures Builder

- Eclipse-Projekt auf Webseite der Vorlesung
  - Evtl. vorinstalliert
  - Eingabe von Gesten
  - Abspeichern (automatisch)

- Gestennamen (für Aufgabe)
  - Offensiv: „air“, „earth“, „fire“
    - greifen Gegner an
  - Defensiv: „defair“, „defearth“, „deffire“
    - währen Angriffsgesten ab



Design der Gesten

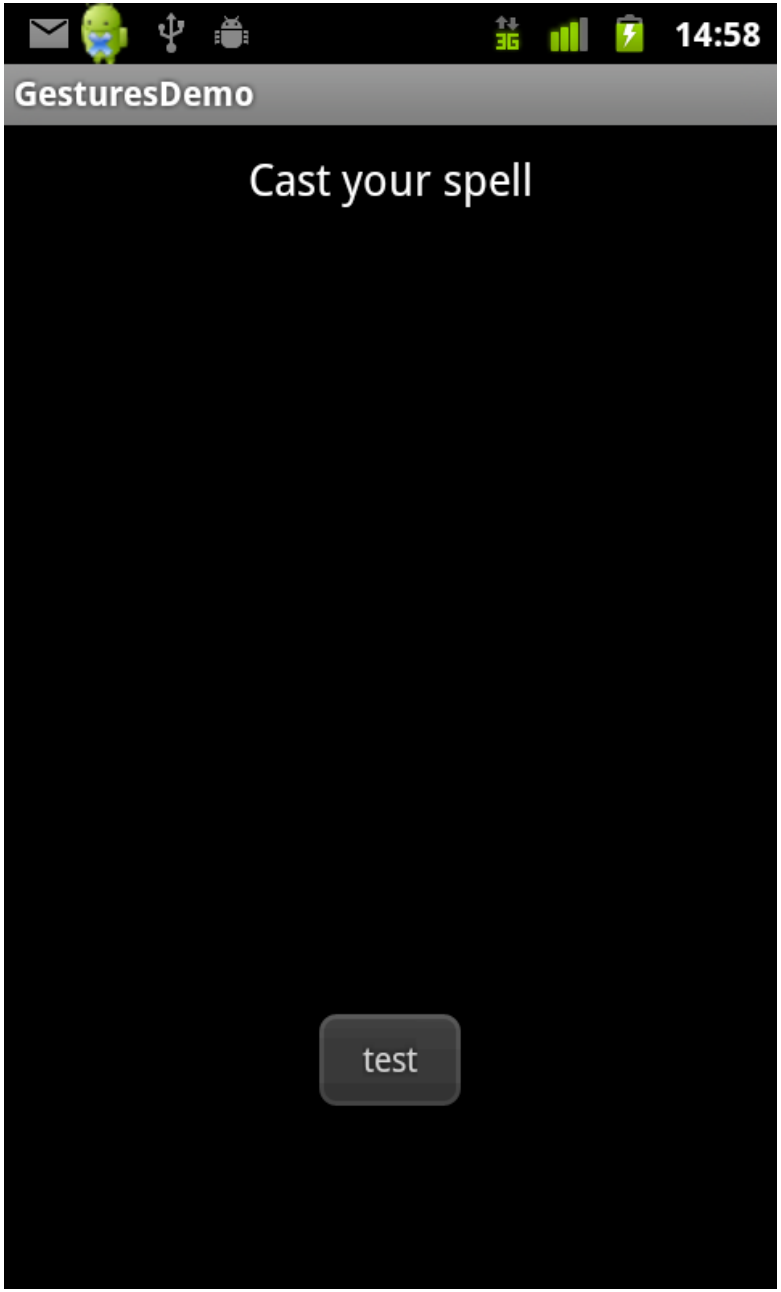
- Wichtig für Benutzer
  - schnell ausführbar (ein oder wenige Strokes, einfache Form)
  - Benutzer kann sich Geste leicht merken
  - Benutzer kann Geste leicht erlernen (motorisch)
  - Geste hat einen Bezug zu ihrem Effekt (z.B. Metapher)
  - hinreichend verschieden (Verwechslungsgefahr)
- Wichtig für Gesten-Erkenner
  - hinreichend verschieden (Verwechslungsgefahr)
- weitere Aspekte?

Benutzer-definierte Gesten

- Problem: Welche Geste für welche Aktion?
  - Verschiedene Benutzer bevorzugen verschiedene Gesten
  - Designer-definierte Gesten oft wenig intuitiv für Benutzer
- Alternative: Benutzer definieren Gesten-Set selbst
  - Designer erklärt Szenario und zeigt Effekt
  - Bittet Benutzer, sich passende Geste auszudenken
  - Beispiel: Effekt: „Air“-Zauberspruch im Spiel, Geste = ?
  - Designer konsolidiert die Vorschläge der Benutzer, achtet auf Randbedingungen (→ vorige Folie)
- Aufgabe: 3 Benutzer nach den 6 Gesten fragen
  - Gesten-Traces auf Papier zeichnen lassen
  - Vorschläge der 3 Benutzer und resultierendes Gesten-Set als PDF
  - kurze Begründung für Aufgreifen/Abweichen von Vorschlägen

Gesture Demo

- Eclipse-Projekt auf Webseite der Vorlesung
  - Eingabe von Gesten
  - Erkennung und Ausgabe des Gestennamens



Android Gestenerkenner

- Package: android.gesture
- Klassen:
  - GestureOverlayView
  - OnGestureListener
  - Gesture
  - Prediction
  - GestureLibrary
  - …

- developer.android.com/reference/android/gesture/package-summary.html

Gesture Overlay View

- GestureOverlayView für Gesteneingaben
  - eigene Schicht über anderen Widgets
  - zeigt Gesten-Trace an
  - verarbeitet Gesteneingaben
  - GestureListener zum GestureOverlayView hinzufügen

- Definition in /res/layout/<name>.xml

```
<android.gesture.GestureOverlayView
  android:id="@+id/gestures_overlay"
  android:layout_width="match_parent"
  android:layout_height="0dip"
  android:layout_weight="1.0"
  android:gestureStrokeType="multiple" />
```

Tipp: Ctrl+Space mit Cursor im XML-Element zeigt mögliche Attribute an

eingegebene Geste verarbeiten

- GestureOverlayView.OnGestureListener registrieren

```
GestureOverlayView overlay = (GestureOverlayView)
                        findViewById(R.id.gestures_overlay);
overlay.addOnGestureListener(new GesturesProcessor());
```

- Eingabe verarbeiten

```
private class GesturesProcessor implements
                        GestureOverlayView.OnGestureListener {
        nächste Folie…
}
```

eingegebene Geste verarbeiten

```
public void onGestureStarted(GestureOverlayView overlay,
                             MotionEvent event) { … }

public void onGesture(GestureOverlayView overlay,
                      MotionEvent event) { … }

public void onGestureEnded(GestureOverlayView overlay,
                           MotionEvent event) {
        gesture = overlay.getGesture();
        …
}

public void onGestureCancelled(GestureOverlayView overlay,
                               MotionEvent event) { … }
```

Einlesen der Gestendaten

```
public class GesturesActivity extends Activity implements
                                                    OnGesturePerformedListener {

private GestureLibrary library;

private final File storeFile = new
                    File(Environment.getExternalStorageDirectory(), "gestures");

public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   library = GestureLibraries.fromFile(storeFile);
   GestureOverlayView gov = (GestureOverlayView) findViewById(R.id.gestures);
   gov.addOnGesturePerformedListener(this);
}
```

Erkannte Geste verarbeiten

```
public void onGesturePerformed(GestureOverlayView o, Gesture g) {
    ArrayList<Prediction> predictions = library.recognize(gesture);
    if (predictions.size() > 0) {
        Prediction prediction = predictions.get(0);
        if (prediction.score > 1.0) {
            // Show the gesture name
            Toast.makeText(this, prediction.name, Toast.LENGTH_SHORT).show();
        }
    }
}
```

- Gesture: gezeichnete Form
  - eine Geste besteht aus einem oder mehreren Strokes
- Prediction enthält „name“ und „score“
  - Name des ähnlichsten Templates
  - Score für die Ähnlichkeit der Eingabe mit dem Template (größer ist ähnlicher)

Spielbildschirm (Beispiel)

Angriff und Verteidigung

Spielregeln

- Offensive und defensive Zaubersprüche
- Gesteneingabe = Zauberspruch sagen

- Offensive Zauber: „air“, „earth“, „fire“
  - wandern über 4s vom einen zum anderen Zauberer
  - Abwehr durch defensive Zaubersprüche **des selben Typs**
    - Beispiel: „fire“ Angriff durch „deffire“ abgewehrt
- Defensive Zaubersprüche: „defair“, „defearth“, „deffire“
  - schützende Aura um Zauberer herum

- für jeden Zauberspruch eine Geste
  - 6 Gesten insgesamt

Spielregeln

- Initial hat jeder Zauberer 10 Punkte
  - -1 Punkte, für nicht abgewehrte Angriffe
  - 0 Punkte = verloren
- Schützende Aura
  - immer nur maximal eine schützende Aura
  - für begrenzte Zeit (5s) aktiv
  - offensiver Zauberspruch zerstört eigene schützende Aura

Aufgabenteile

- Programmgerüst herunterladen
- Bildressourcen erstellen
  - Hintergrundbildschirm
  - Zauberer 1
  - Zauberer 2
  - drei offensive Zaubersprüche
  - drei defensive Zaubersprüche

  - Template-png-Dateien verwenden
  - Größen einhalten
- Benutzer-definierte Gesten
  - 3 Benutzer um Vorschläge für die 6 Gesten bitten
  - Vorschläge zu einem Gesten-Set konsolidieren

Aufgabenteile

- Erstellung von Gesten mit dem GestureBuilder
  - Exportieren (vom Handy/Emulator zum PC) mit:
    `adb pull /mnt/sdcard/gestures`
- Erstellung der Animationen
  - wie im Aufgabenblatt angegeben
  - je Zauberer **eine** Animation für offensive und **eine** Animation für defensive Zaubersprüche
- Verbesserung der Strategie des vom Handy gesteuerten Zauberers
  - momentan zufällige Aktion alle 4s
  - bessere, aber nicht perfekte Strategie
- Ausgabe des Gestenvokabulars
  - Methode toBitmap der Klasse Gesture

Animationen in XML

- Kombination über set-Element

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator" >
    <alpha … />
    <scale … />
    <translate …/>
</set>
```

Kombination von alpha, scale, translate

beschleunigt (gilt für alle Elemente im set)

Animationen in XML

- Transparenz: alpha-Element
- Beispiel: 10x von alpha = 0.5 auf alpha = 1.0, jeweils 500ms, erst vorwärts dann rückwärts abspielen

```
<alpha
    android:duration="500"
    android:fromAlpha="0.5"
    android:repeatCount="10"
    android:repeatMode="reverse"
    android:toAlpha="1" />
```

Animationen in XML

- Skalieren: scale-Element
- Beispiel: In 300 ms von 30% auf 80% der Originalgröße skalieren. Um den Mittelpunkt (PivotX,Y = 50%,50%) skalieren.

```
<scale
    android:duration="300"
    android:fromXScale="0.3"
    android:fromYScale="0.3"
    android:pivotX= 50%"
    android:pivotY="50%"
    android:toXScale="0.8"
    android:toYScale="0.8" />
```

Animationen in XML

- Skalieren: scale-Element
- Beispiel: In 300 ms von 30% auf 80% der Originalgröße skalieren. Um den Mittelpunkt (PivotX,Y = 50%,50%) skalieren. Zunächst 1s warten (relativ zu anderen im set).

```
<scale
    android:duration="300"
    android:startOffset= "1000"
    android:fromXScale="0.3"
    android:fromYScale="0.3"
    android:pivotX= 50%"
    android:pivotY="50%"
    android:toXScale="0.8"
    android:toYScale="0.8" />
```

Startzeit 1s nach Start der set-Animation

Drehung um diesen Punkt (Pivot-Punkt)

Animationen in XML

- Translation: translate-Element
- Beispiel: In 2s in x-Richtung von 400 auf 800 Pixel Offset bewegen. Anfangs- und Endzustand nach Animation bestehen lassen.

```
<translate
    android:duration="2000"
    android:fromXDelta="400"
    android:toXDelta="800"
    android:fillEnabled="true"
    android:fillBefore="true"
    android:fillAfter="true"
    />
```

fill...: Anfangs- und Endzustand (400 bzw. 800 Pixel offset) vor bzw. nach der Animation bestehen lassen.

Animationen in XML

- Rotation: rotate-Element
- Beispiel: In 4s um den Punkt (50%,40%) (relativ zur linken oberen Ecke und zur Größe des Bildes) von 0° auf 400° drehen.

```
<rotate
    android:duration="4000"
    android:fromDegrees="0"
    android:pivotX="50%"
    android:pivotY="40%"
    android:toDegrees="400" />
```

Abgabe

- Plagiate sind verboten und führen zum Ausschluss aus der Veranstaltung!
- Dieses Übungsblatt muss einzeln bearbeitet werden.
- Bestandteile der Abgabe
  - Aus Eclipse exportiertes Projekt (Export → Archive file, zip-Format)
  - PDF-Datei für Aufgabenteil (b)
  - gestures-Datei für Aufgabenteil (c)
- Abgabe bis zum 28.11.2011 um 12:00 Uhr im **neuen** UniWorX Portal (https://uniworx.ifi.lmu.de/) ab.
- Sie sollten Ihre Lösung in der Übung vorstellen können!