Dokumentation iPad Spot The Difference

Objective-C/Cocoa Vorlesung

Bericht von

Andreas Knöpfle (281187)

Mathias Hodler(281159)

Konstanz, 2. März 2011

Inhaltsverzeichnis

1 Abkürzungsverzeichniss

JPEG Joint Photographic Experts Group, ein im Web weit verbreitetes Grafikformat für verlustbehaftete oder verlustfreie Kompression von digitalen Fotografien.

IOS ein Betriebssystem der Firma Apple für mobile Geräte. Es basiert auf Mac OS X und ist das Standard-Betriebssystem der Apple-Produkte iPhone, iPod touch, iPad und der zweiten Generation des Apple TV

plist Property Lists - speichern Einstellungen in Mac OS X

2 Idee und Spielkonzept

Im Rahmen des Wahlpflichtfaches "Objective-C/Cocoa" musste eine Anwendung für das iPhone bzw. iPad erstellt werden. Dabei sollte auch wenn möglich die Besonderheiten von IOS wie z.B. der Lagesensor, GPS und Touch-Funktionen mitverwendet werden.

Wir haben uns Entschlossen eine neue Art Fehlersuchspiel zu entwickeln. Dieses lehnt sich an die bekannten Fehlersuchspiele in Zeitschriften an, bei denen zwei auf den ersten Blick identische Bilder nebeneinander zu sehen sind, wobei eines davon mehrere Veränderungen enthält, welche zu finden sind (siehe Abbildung 1). In "iPad Spot The Difference" liegt jedoch die Besonderheit, dass das Fehlerbild ein Foto aus der Realität ist und um die Fehler zu finden, dass auf dem Display dargestellten Foto auch mit der realen Umgebung verglichen werden muss.

Abbildung 1: Typisches Fehlersuchbild in Zeitschriften

Der Spieler erhält zu Beginn eine Weltkarte angezeigt, auf der alle Punkte markiert sind, für diese ein Fehlerbild existiert. Befindet sich der Spieler in der örtlichen Nähe einer solchen Markierung kann er das Fehlerbild mit der realen Umgebung vergleichen und die Fehler durch einen Klick auf das Bild aufdecken.

Das Spiel wurde speziell für das iPad entwickelt, da eine möglichst großes Display nötig ist, um auch die Fehler zu erkennen.

3 Spezifikation

Im folgenden werden die verschiedenen Benutzeroberflächen gezeigt und die grundlegende Architektur festgelegt.

3.1 Benutzerschnittstellen

Die Benutzerschnittstelle beinhaltet folgende Ansichten:

- Kartenansicht
- Fehlerbildansicht
- Hilfeansicht

3.1.1 Kartenansicht

In der Kartenansicht 2 werden dem Benutzer alle Fehlerbilder als Pin auf einer Weltkarte angezeigt. Er hat die Möglichkeit ein Fehlerbild auszuwählen. Dieses wird ihm dann als Miniaturbild mit der Anzahl der darin enthaltenen Fehlern und des Titels angezeigt.

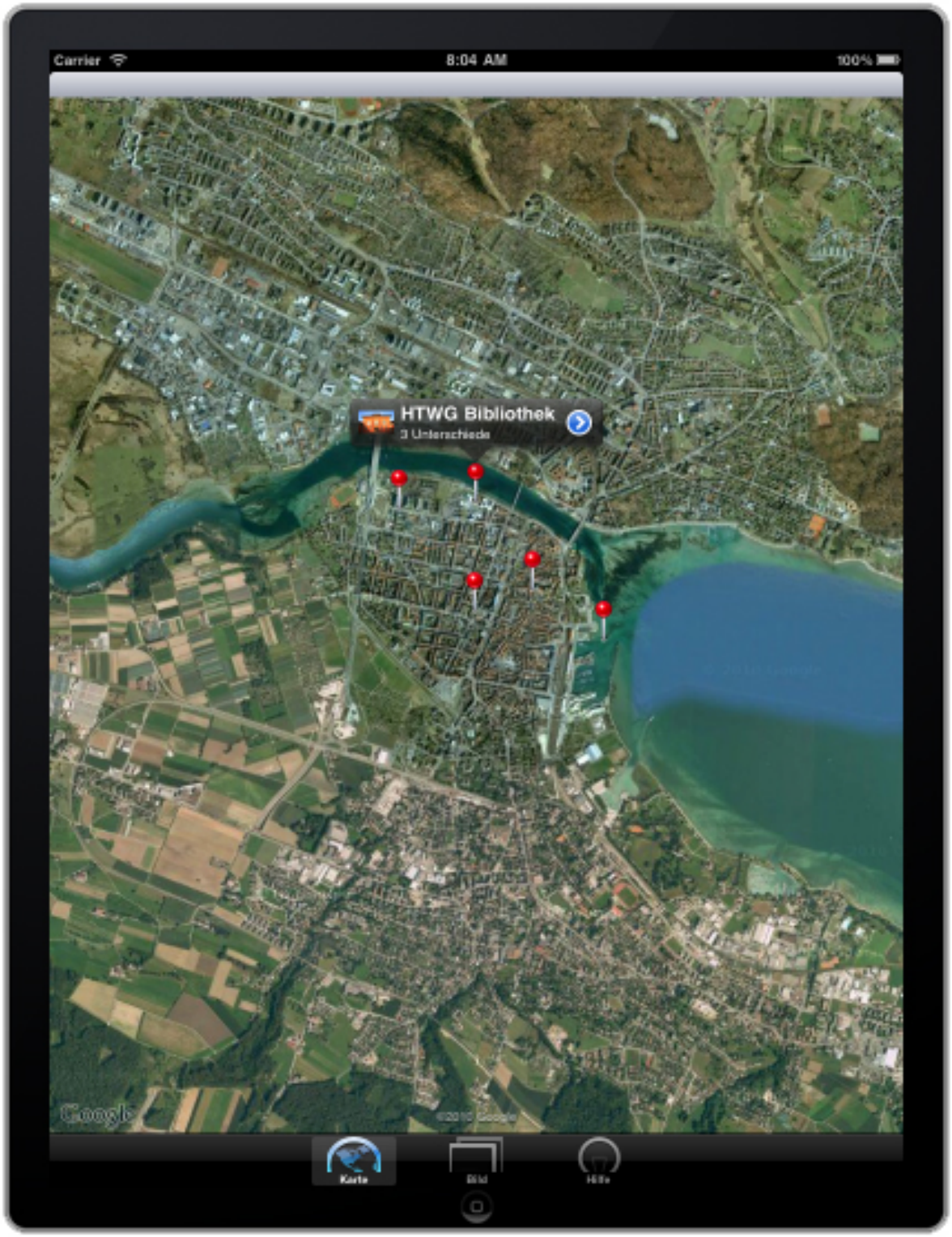


Abbildung 2: Kartenansicht

3.1.2 Bildansicht

In der Bildansicht 3 wird dem Benutzer das fehlerhafte Bild angezeigt. Durch die bereits aus anderen Anwendung bekannte Zoom-Geste kann das Bild vergrößert und verkleinert werden. Durch langes Drücken auf die Stelle an der der Fehler ist wird der Bereich grün eingefärbt und der Fehler wird gewertet. Ein langes Drücken auf eine andere Stelle bewirkt nichts. Durch eine Schüttelgeste wird ein Rahmen auf dem Bild gezeichnet, der einen Bereich mit einem Fehler umschließt 5.

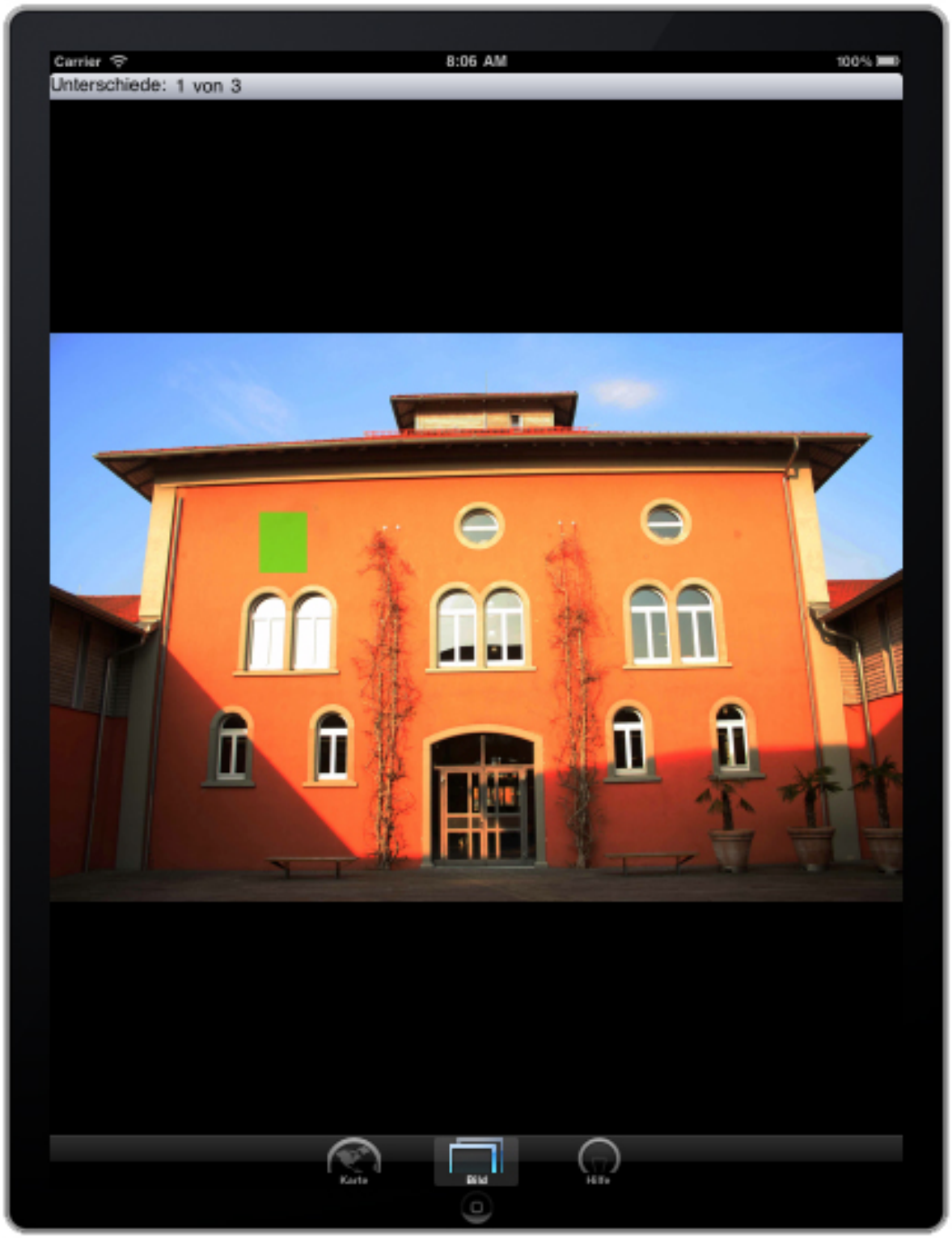


Abbildung 3: Bildansicht - normal

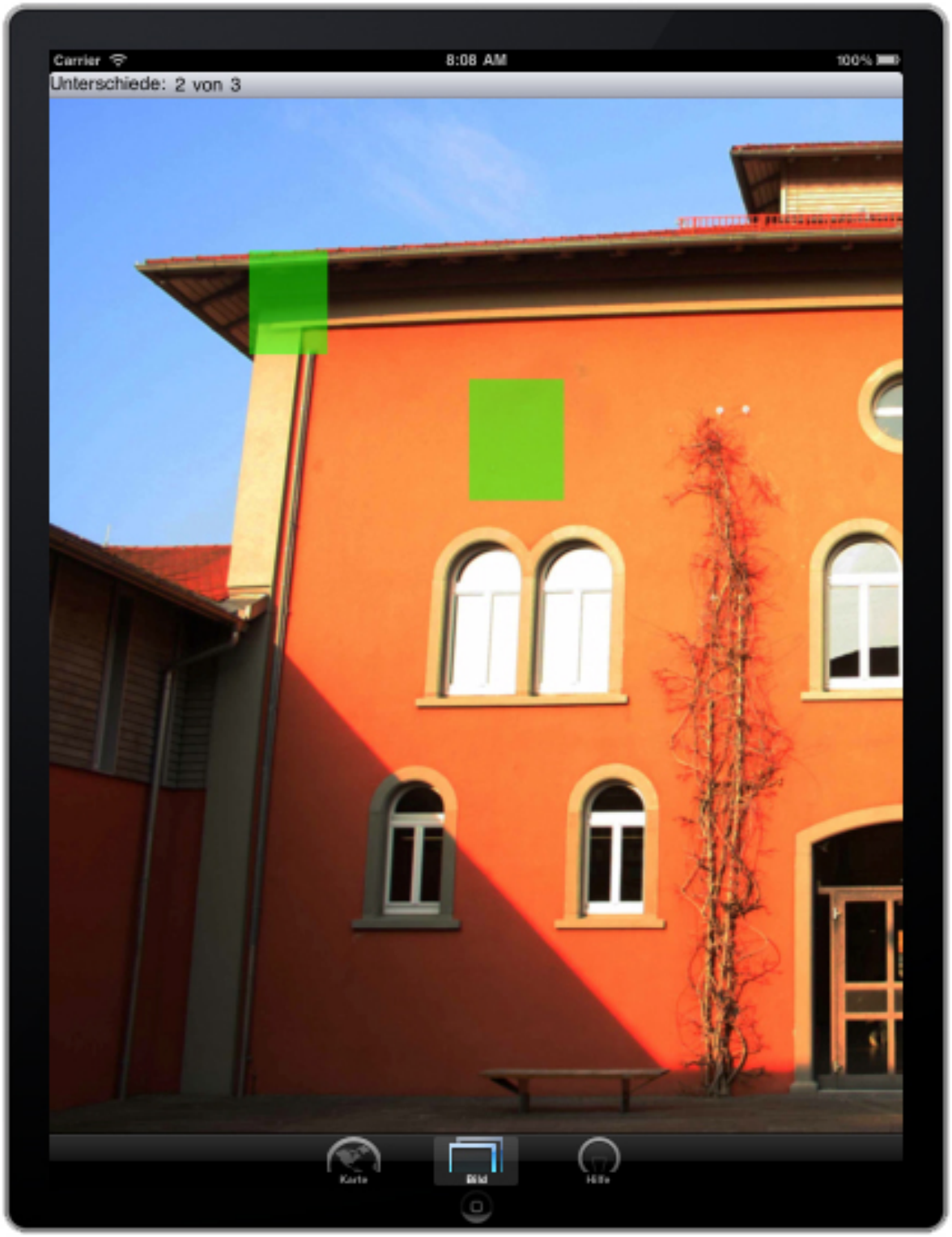


Abbildung 4: Bildansicht - vergrößertes Bild

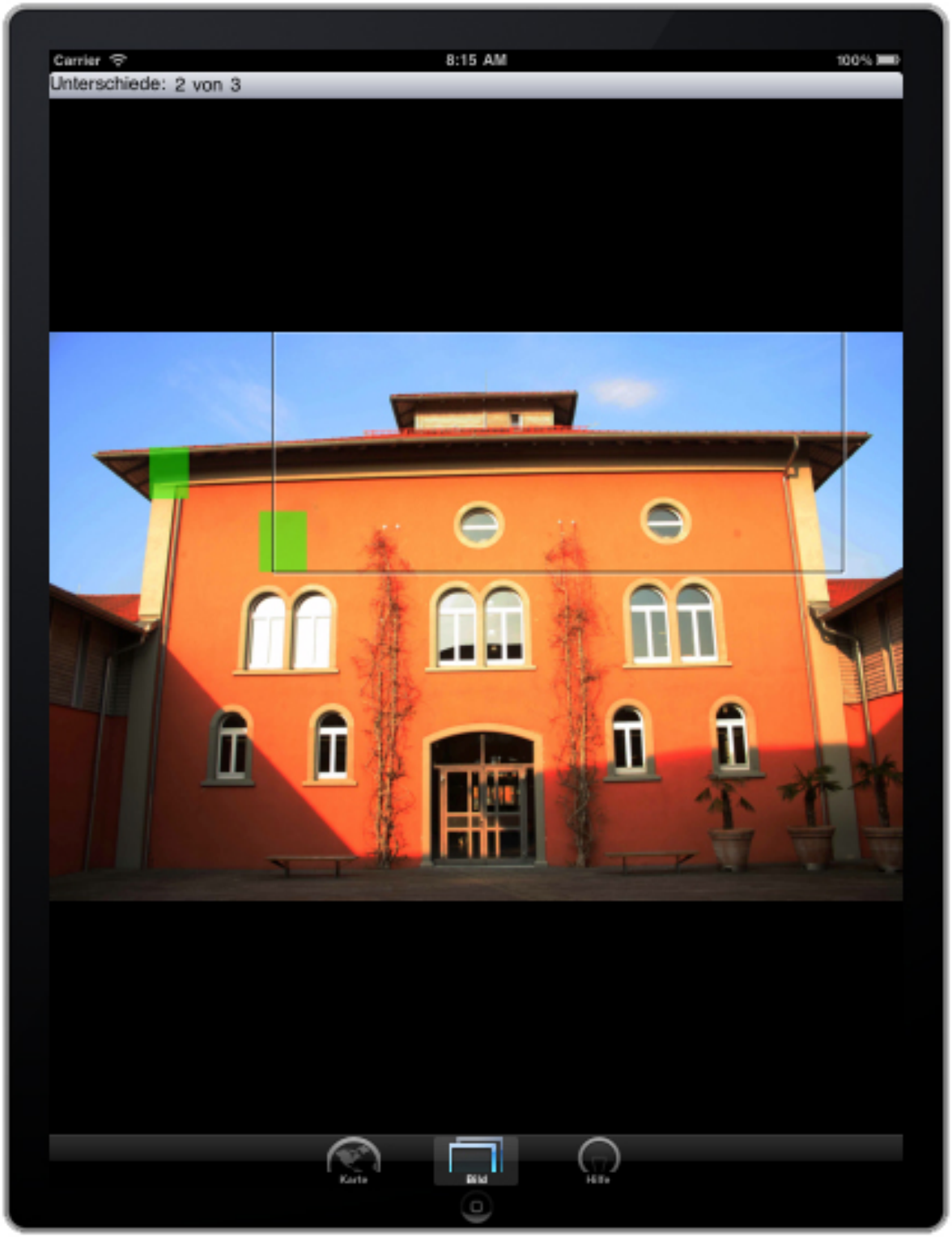


Abbildung 5: Bildansicht - Fehlerhinweis

3.2 Architektur

Das Klassendiagramm (siehe Abbildung 6) zeigt den Aufbau der Anwendung. Die einzelnen Klassen und deren Funktionen sind im folgenden erklärt.

ImageManager

Die Klasse ImageManager hat die Aufgabe aus einer Datenquelle alle SpotImages zu erzeugen und diese zu verwalten.

SpotImage

Ein SpotImage definiert ein spezielles Fehlerbild mitsamt seinen geographischen Koordinaten, Titel, Beschreibungstext und den eigentlichen Fehlern (Klasse Difference). Sie bietet über die Funktion "doesHitWith" die Möglichkeit anhand der Angabe einer xy-Koordinate festzustellen ob dort ein Fehler im Bild existiert. Diese Funktion wird benötigt wenn der Benutzer auf das Bild, um zu überprüfen ob dort auch wirklich ein Fehler versteckt ist.

Difference

Die Klasse Difference stellt einen einzelnen Fehler in einem SpotImage dar. Dieser wird durch seine Position im Bild (xy-Koordinate) und dessen Größe (Breite und Höhe) definiert.

MapViewController

Der MapViewController ist für den View zuständig, der die Weltkarte und die SpotImages anzeigt.

ImageViewController

Das Anzeigen des im MapViewController ausgewählten SpotImage geschieht im View des ImageViewController. Der ImageViewController besitzt daher eine Referenz auf das

gerade aktive SpotImage. Alle aufgedeckten Fehler werden temporär gemerkt um dem Spieler zu zeigen wie viele Fehler noch zu finden sind.

AboutViewController

Für die Anzeige einer Kurzanleitung für das Spiel ist der AboutViewController zuständig.

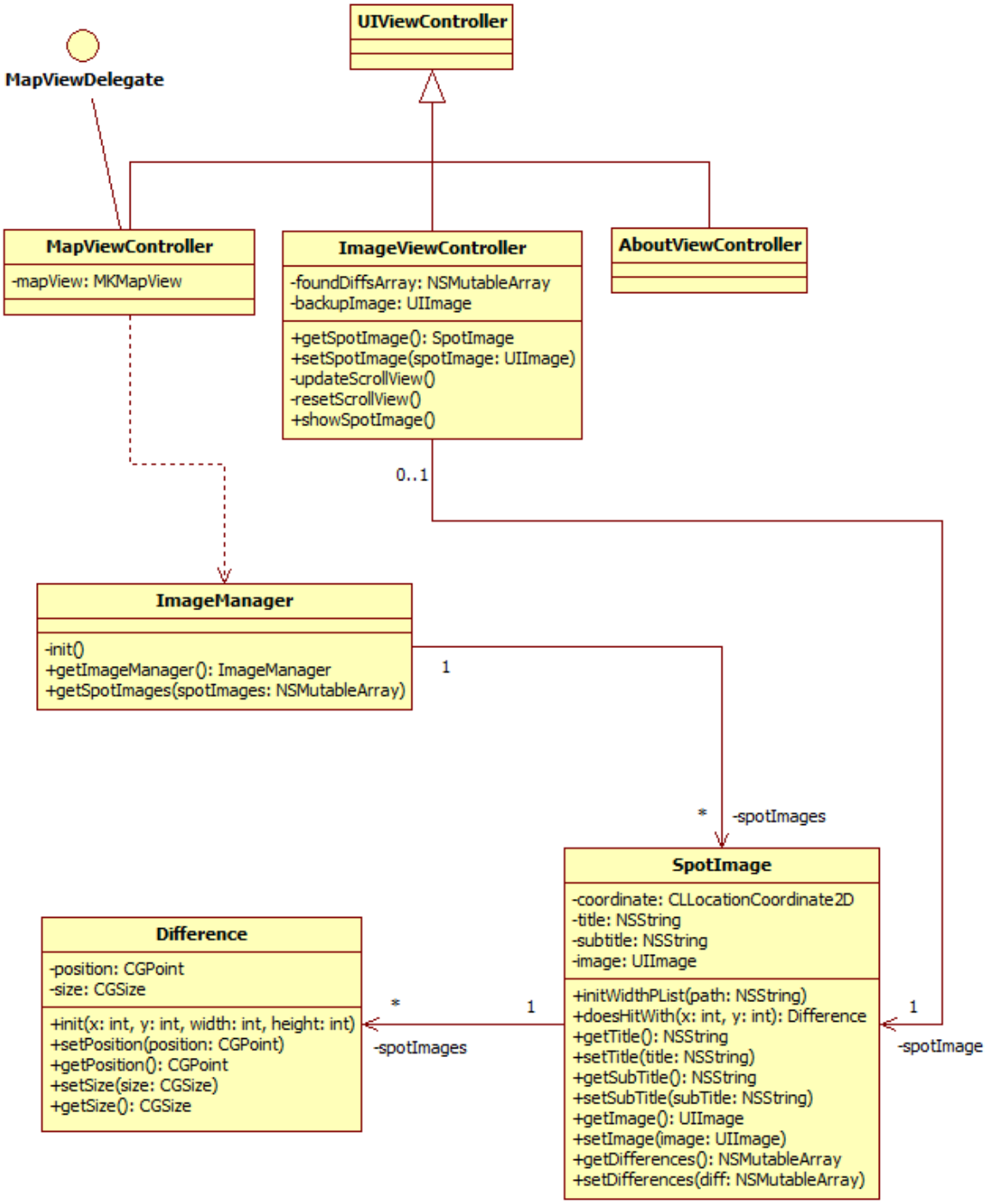


Abbildung 6: UML-Klassendiagramm

4 Umsetzung

Die Umsetzung der verschiedenen Views und Controller werden im folgenden erklärt.

4.1 Struktur der Bild- und Fehlerdaten

Für jedes Fehlerbild ist es nötig den Titel, eine Beschreibung, der geographischen Koordinaten, sowie die Position der Fehler festzuhalten. Gespeichert werden diese Daten pro Fehlerbild in einer separaten Property List im Ordner "Metadata". Jede plist erhält daher noch den Pfad zum dazugehörigen Fehlerbild (im Ordner "Images").
Da ein Fehlerbild mehrere Fehler beinhalten kann, werden diese in einem Array innerhalb der plist definiert. Jeder Fehler ist durch eine xy-Koordinate und einer Höhen- und Breiteninformation definiert.

Listing 1: Ausschnitt aus einer plist

```
<dict>
        <key>Title</key>
        <string>Farbenfroh</string>
        <key>Description</key>
        <string>Europahaus</string>
        <key>ImagePath</key>
        <string>IMG_2693.jpg</string>
        <key>Latitude</key>
        <real>47.66762719262464</real>
        <key>Longitude</key>
        <real>9.164882898330688</real>
        <key>Differences</key>
        <array>
                <dict>
                        <key>X</key>
                        <integer>1320</integer>
                        <key>Y</key>
                        <integer>2610</integer>
                        <key>Width</key>
                        <integer>313</integer>
                        <key>Height</key>
                        <integer>726</integer>
                </dict>
                <dict>
                        <key>X</key>
                        <integer>2340</integer>
                        <key>Y</key>
                        <integer>714</integer>
                        <key>Height</key>
                        <integer>246</integer>
                        <key>Width</key>
                        <integer>300</integer>
```

```
33                  </dict>
34          </array>
35 </dict>
```

Eingelesen werden die plists beim erstmaligen Aufruf der init-Funktion des "ImageManager", welcher als Singleton implementiert ist.

4.2 Kartenansicht

Die Kartenansicht besteht aus einem MKMapView-Element in das für jedes vorhandene SpotImage eine Annotation in der Karte erzeugt wird. Der wichtigste Teil dabei ist für die einzelnen Annotationen jeweils einen MKAnnotationView zu erzeugen, damit sie angezeigt werden können (viewForAnnotation-Methode). In diesem Fall wurde für die Spotimages ein MKPinAnnotationView gewählt, der die Annotationen als Pins auf der Karte anzeigt. Über die sogennanten AccessoryViews im MKPinAnnotationView kann auch definiert werden wie eine ausgewählte Annotation aussehen soll. Hier wird für den leftCalloutAccessoryView, also den View links des Bildtitels, ein Vorschaubild des Fehlerbilds angezeigt. Das Vorschaubild wird durch eine Erweiterung der Klasse UIImage (UIImage+Resize) erzeugt. Für den rightCalloutAccessoryView wird ein Button angezeigt der den Benutzer zum Fehlerbild führt.

4.3 Fehlerbildansicht

In der Fehlerbildansicht wird das gewählte Bild der Kartenansicht angezeigt. Der Spieler kann durch Klicks die gefundenen Fehler in dieser Ansicht aufdecken.

4.3.1 Allgemeiner Aufbau

Für die Fehlerbildansicht war es nötig, dass das Bild vom Spieler vergrößert und verschoben werden kann. Im Gegensatz zu einem statisch an die Bildschirmgröße angepasstes Bild, lassen sich die Fehler besser erkennen. Dieses Problem wurde mittels eines UIScrollView und einem darin platzierten UIImage gelöst. Das Bild lässt sich somit per "Pinch to Zoom" vergrößern oder verkleinern, wie auch im vergrößerten Zustand verschieben.

Bei der automatischen Anpassung des Bildes mittels des UIScrollView gibt es aber einige Darstellungsprobleme. So wird das Bild zu beginn bzw. wenn dieses kleiner ist als der Bildschirm, nicht innerhalb der UIScrollView zentriert. Dazu wird per Delegate bei jedem Zoom der UIScrollView (scrollViewDidZoom) die Offsets für ein zentriertes Bild neu berechnet.
Ein weiteres Problem stellen unterschiedlich Große Bilder, wie auch verschiedene Seitenverältnisse dar. Besitzt die UIScrollView nicht die selben Maße wie das Bild, so entstehen schwarze Ränder außerhalb des Bildes, oder das Bild wird abgeschnitten. Um dies zu verhindern, wird beim Anzeigen des Bildes (showSpotImage) das UIScrollView entsprechend angepasst.

4.3.2 Fehlererkennung

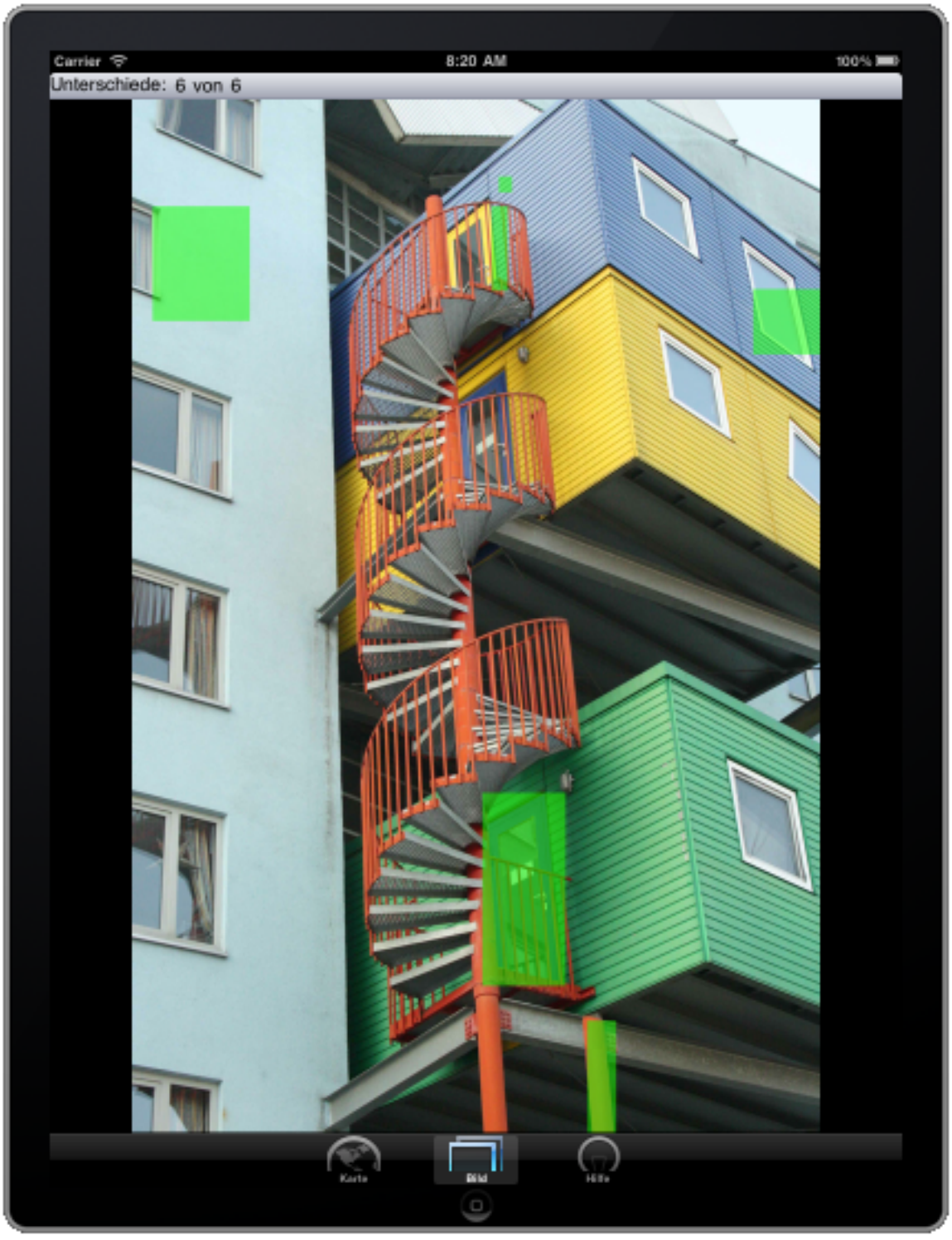
Um einen Fehler aufzudecken, muss der Spieler einen langen Klick auf die jeweilige Stelle auf dem Bild tätigen. Bei einem einfachen Klick wäre die Gefahr zu größ gewesen, dass ausversehen Klicks ausgeführt werden.
Da aber die Möglichkeit besteht das Bild zu vergrößern und zu verschieben, kann man die erhaltenen Koordinaten des Klicks nicht 1:1 übernehmen. Die Koordinaten beziehen sich nämlich auf die aktuelle Größe des Bildes. So kann bei einem verkleinerten Bild die rechte untere Ecke die Koordinaten (100,100) besitzen, bei einer starken Vergrößerung jedoch (500,500). Daher muss diese Koordinaten in der Funktion longPress anhand dem Verhältnis der dargestellten Bildgröße und der originalen Bildgröße korrigiert werden.
Für die eigentliche Fehlererkennung, wird die Funktion "doesHitWithXandY" des dargstellten SpotImage aufgerufen. Durch die Übergabe der korrigierten Klick-Koordinate erhält man ein boolschen Wert zurück, der angibt, ob man einen Fehler getroffen hat.
Danach wird ggf. gefundene Fehler im Bild mit einem Rechteck markiert und Anzahl der gefundenen Fehler inkrementiert.

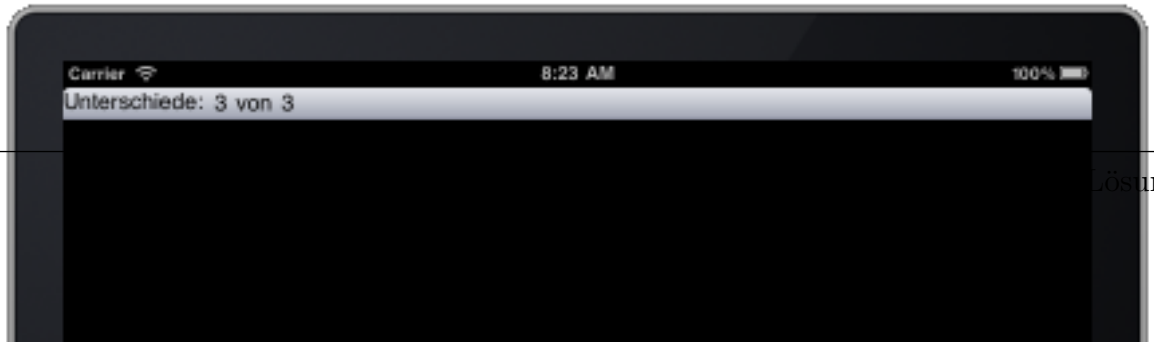
4.3.3 Hilfestellung

Da es auch mal vorkomme kann, dass der Spieler auch nach langem Suchen die Fehler nicht findet, so wird durch Schütteln des iPads eine Hilfestellung angeboten. Um einen der Fehler wird dazu ein Rahmen mit zufälliger Größe gespannt, der den Fehler eingrenzt.

5 Lösung

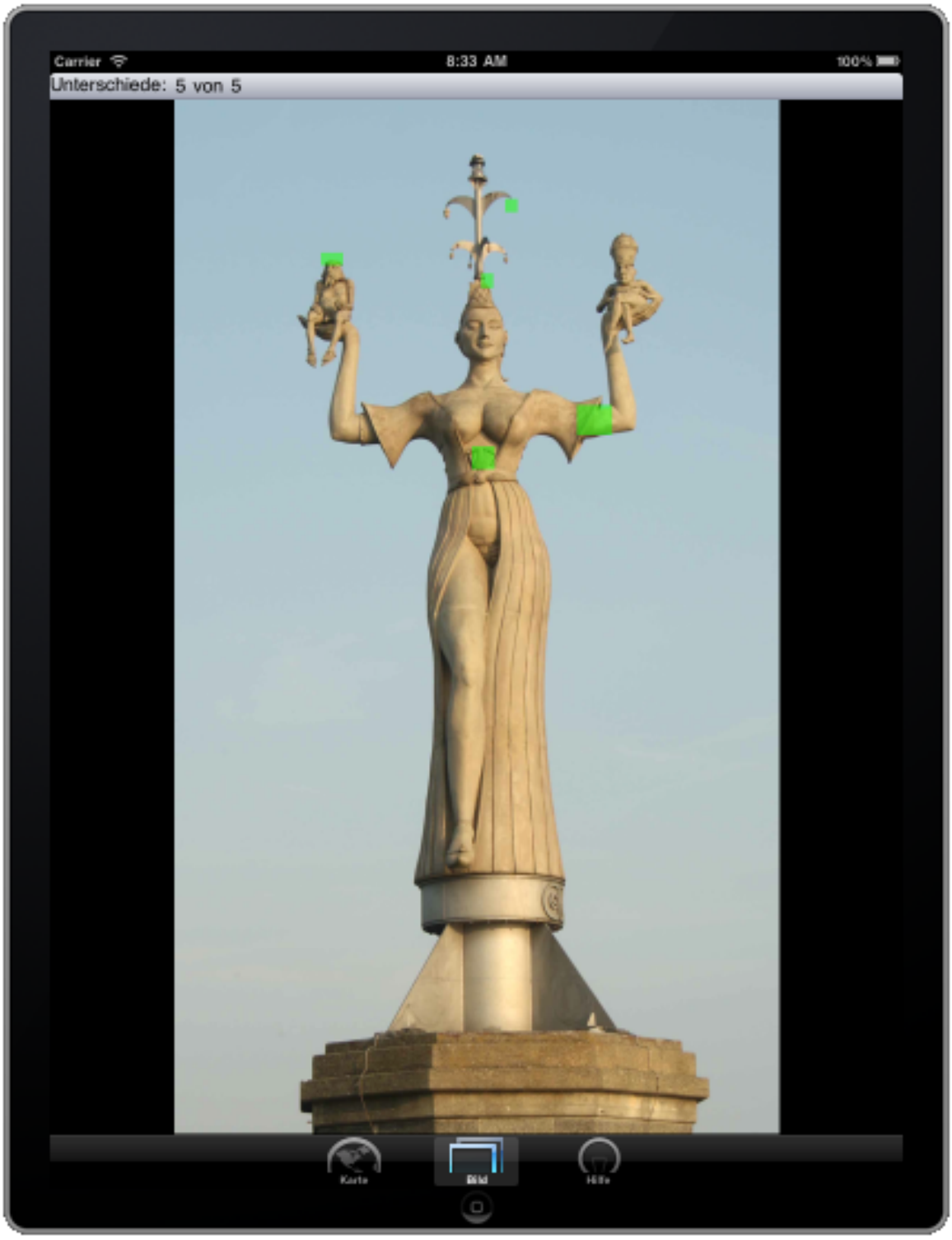
Im folgenden Abschnitt sind die Lösungen zu den Rätseln abgebildet.


Carrier
8:20 AM
100%
Unterschiede: 6 von 6
Karte
Bild
Hilfe


Carrier
8:23 AM
100%
Unterschiede: 3 von 3


Carrier
8:24 AM
100%
Unterschiede: 5 von 5
Karte
Bild
Hilfe


Carrier
8:32 AM
100%
Unterschiede: 5 von 5


Carrier
8:33 AM
100%
Unterschiede: 5 von 5
Karte
Bild
Hilfe

Literatur und Quellenverzeichnis

[1] Titel, Autor

http://www.example.com

Stand: 17.03.2010