Übung 1 – Hello Cocoa – per Hand

1 Aufgabenstellung

Schreibt die *Hello Cocoa* Anwendung, die wir gerade mit dem Interface Builder erstellt haben, von Hand nach. Achtet auf den Speicher den ihr anfordert und ob ihr ihn wieder freigebt.

2 Zusatzaufgaben

1. Ändert die Schriftart und Schriftgröße des Labels.
2. Ändert den Titel und die Schriftfarbe des Button wenn dieser gedrückt ist auf eine Farbe eurer Wahl mit dem Titel „touched“.

3 Hinweise

Benutzt nicht den Interface Builder!
Benutzt die Dokumentation.

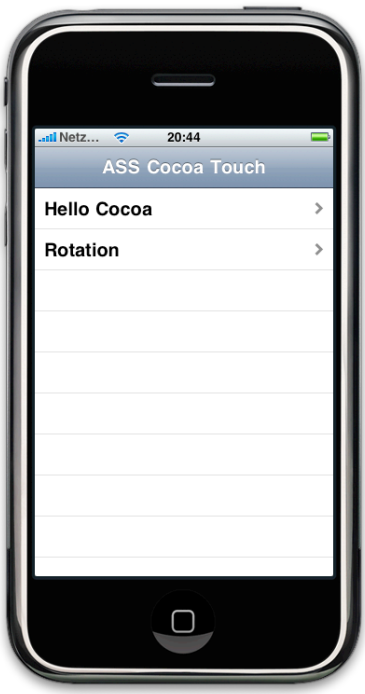
Tag 2 – UINavigationController und UITableView

Erstellt ein neues Projekt. Benutzt dafür das *Navigation-based Application* Template. Dieses Template enhält schon einen NavigationController in dem ein UITableViewController angezeigt wird. Kopiert die ViewController aus der Übung vom ersten Tag (HelloCocoa) und der Rotations-Übung in das neue Projekt.

Aufgabe 1

Ziel der ersten Aufgabe ist es den TableView für jeden der zuvor kopierten ViewController eine Zeile anzeigen zu lassen, in der der Name der Übung steht. Als Datenmodell soll ein Array verwendet werden. Erstellt dafür in *viewDidLoad* ein *NSArray* (mit dem Convinient Constructor *arrayWithObjects:*), das die anzuzeigenden Zellen-Titel enthält. Achtet dabei auf das Speichermanagement.

Damit erkennbar ist, dass die Zellen antippbar sind und dadurch eine DrillDown-Animation ausgelößt wird, sollen die Tabellen-Zellen rechts den entsprechenden AccessoryView anzeigen. Setzt außerdem einen Titel in der NavigationBar.



Aufgabe 2

Jetzt soll bei der Berührung einer Zelle der entsprechende ViewController auf den NavigationController gepusht werden. Implementiert dazu die richtige Methode des UITableViewDataSource bzw. UITableViewDelegate Protokolls. Gebt den ViewControllern der HelloCocoa und Rotations Übungen noch einen Titel (falls noch nicht geschehen), damit die NavigationBar diesen anzeigen kann.

Tag 2 – Touches und Animationen

1 Aufgabenstellung

1.1 Touches

Erstellt eine neue *view-based application* Namens *Touches*. Fügt in das Projekt das Bild „monster.png“ ein. Positioniert das Monster mit dem InterfaceBuilder an Position x=20 y=20 mit einer Breite und Höhe von 80 Pixel. Schreibt ein Outlet für das Monster.

Der zugehörige ViewController soll folgendes können:

- Wenn das Monster berührt wird soll es dem Finger auf dem Bildschirm folgen.
- Wenn auf dem Schirm ein *DoubleTap* ausgeführt wird, soll das Monster wieder an der Position 20, 20 positioniert werden.

1.2 Animationen

Es soll der Umgang mit einfachen Animationen eingeübt. Es soll, wie in der Übung zu den Touches, mit Hilfe des Interface Builders ein Bild (z.B. ein Monster) in einem *UIImageView* auf einem schwarzen Hintergrund angezeigt werden. Außerdem muss wieder ein Outlet in Eurem ViewController auf den *UIImageView* mit dem Bild Eurer Wahl zeigen.

1. Zuerst wird das Monster nur bewegt. Sobald man auf den Bildschirm tippt soll sich das Monster(animiert) zu dieser Position bewegen. Implementiert dazu die Animation in der Methode *touchesBegan:withEvent:* Eures *UIViewControllers*.

2. Als nächstes soll das Monster, sobald die Bewegungs-Animation zu Ende ist, zweimal ein- und ausgeblendet werden. Implementiert dazu die Methode *animationDidStop:finished:context:*, in der Ihr den *alpha*-Wert des Monsters animiert. Damit sich die Animation wiederholt muss der repeatCount auf zwei gesetzt werden. Die Standard-Dauer einer Animation ist zu lange. Setzt die Dauer der Animation für das Ein- und Ausblenden auf 0.2 Sekunden. Damit das Monster nach dem Ein- und Ausblenden wieder sichtbar wird, muss der alpha-Wert nach dem Ende der alpha-Animation explizit wieder auf 1.0 gesetzt werden.

2 Hinweise

Benutzt *touchesBegan* und *touchesMoved* der Klasse *UIResponder* von der alle UIViewController abgeleitet sind.

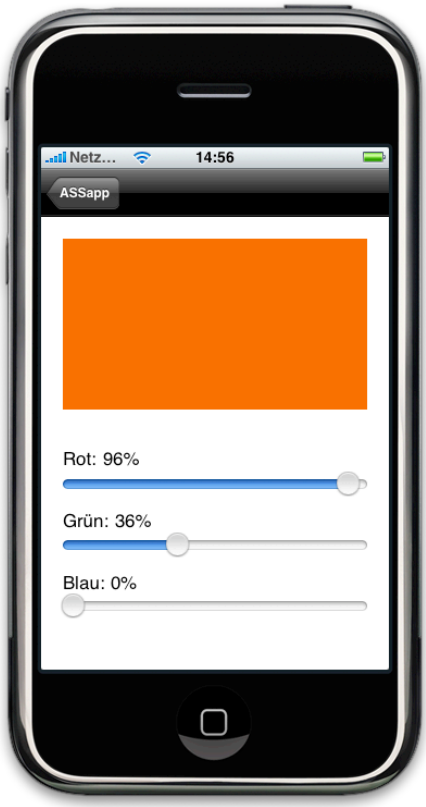
Die Position eines touches kann man z.B. so auslesen: CGPoint point = [touch locationInView:self.view];

Ob sich ein Punkt in einem gegebenen Rechteck befindet kann mit der Methode *CGRectContainsPoint* ausgelesen werden.

Tag 2 – Rotation

1 Aufgabenstellung

Erstellt ein neues Projekt und bringt in seinem XIB ein View, drei Label und drei Slider an. Diese sollen auf die Rotation des Gerätes reagieren und sich anpassen. Wenn die Slider in ihrem Wert verändert werden, soll sich die Farbe der View anpassen — RGB). Die Label zeigen den Prozentwert des jeweils gewählten Farbwertes an.



2 Hinweise

Tag 3 – HTTPConnection

1 Aufgabenstellung

In dieser Aufgabe sollen die Tweets von beliebiegen öffentlichen Twitter-Accounts abgefragt und angezeigt werden können. Erstellt dazu eine neue *Navigation-based Application*. Falls ihr diese Aufgabe in einem bestehenden Projekt lösen möchtet, könnt ihr auch einen neuen UITableViewController erstellen.

Fügt eine UISearchBar zum View des RootViewControllers hinzu, sodass sie direkt über dem UITableView liegt. In dieser SearchBar soll später der Name des Accounts eingegeben werden, dessen Tweeds abgefragt werden.

Sorgt dafür, dass der *Cancel*-Button angezeigt wird. Wenn der *Cancel*-Button oder der *Search*-Button (auf dem Keyboard) gedrückt wird soll die Tastatur verschwinden. Falls in der SearchBar ein Text eingegeben wurde sollen die Tweets des eingegebenen Accounts abgefragt werden.

Benutzt dazu die URL: `http://api.twitter.com/1/statuses/user_timeline/%@.xml` und ersetzt das %@ durch den Text aus der SearchBar. Die Abfrage soll asynchron sein!

Das Ergebnis der Abfrage könnt ihr an einen von uns geschriebenen Parser senden. Kopiert den Parser in euer Projekt und implementiert seinen Delegate. Der Parser liefert ein Array aus Strings mit den Texten zurück.

2 Zusatzaufgaben

2.1

Die Standardzellen sind einzeilig und die Tweets passen somit nicht ganz hinein. Bearbeitet das Label der Standardzelle so, dass der ganze Text sichtbar ist. Dazu kann die Schriftgröße und/oder die Anzahl der Zeilen des Labels geändert werden. Eine weitere Möglichkeit ist es die Höhe der Zellen anzupassen.

2.2

Lest in der Dokumentation nach, wie Custom-TableViewCells erstellt werden können. Der Artikel dazu heisst: *A Closer Look at Table-View Cells*. Erstellt Euer eigenes Zellen-Design im Interface Builder für die Twitter App.

3 Hinweise

Es gibt einen ausführlichen Artikel zu NSURLConnection mit Beispiel-Code in der Dokumentation unter *Using NSURLConnection*. Um ihn zu finden müsst Ihr die Volltext-Suche aktivieren.

Unter `http://apiwiki.twitter.com/Twitter-API-Documentation` gibt es eine ausführliche Dokumentation der Twitter-API.

Tag 3 – UIPicker

1 Aufgabenstellung

Erstellt ein neues Projekt für eine „PickerView“ und fügt in dieses zwei Label und einen Picker hinzu. Das untere Label und der Picker sollen als Outlet der Klasse bekannt gemacht werden.

Die Klasse soll die Protokolle *UIPickerViewDelegate* und *UIPickerViewDataSource* benutzen. Die Zeilen des Pickers sollen die Wochentage von Montag bis Freitag beinhalten, das untere Label soll hierzu das aktuelle (TOP)Gericht der Mensa anzeigen. Wenn ein anderer Tag ausgewählt wird, soll sich das Label aktualisieren.

Speichert welcher Tag ausgewählt ist in den UserDefault und ladet den ausgewählten Tag beim nächsten Start.

2 Hinweise

Um den Picker sichtbar zu machen muß der Delegate und die DataSource gesetzt werden. (Macht dieses am besten gleich zu Beginn).

Tag 3 – CoreData

1 Aufgabenstellung

Erstellt ein neues Projekt namens *Memo*. Benutzt dafür das *Navigation-based Application* Template und setzt einen Haken bei der Option *Use Core Data for Storage*. Dies erstellt Euch das bekannte *Navigation-based Application* Template und fügt eine ganze Reihe von Funktionen hinzu, die es schon zu einer vollwertigen App machen, mit der Daten abgespeichert und gelöscht werden können.

In dieser Übung wird das Template zu einer Memo-Anwendung umgebaut, die es erlaubt, Memos mit einem Text und einem Datum abzuspeichern, zu suchen und einzelne oder alle Memos zu löschen.

1.1 Benutzerschnittstelle

Zuerst soll die Benutzerschnittstelle vorbereitet werden. Die NavigationBar soll zwei UIBarButtonItems enthalten - einen *Edit*- und einen *Alles löschen*-Button. Die Zellen des UITableViews sollen so aussehen wie in Abbildung 1, benutzt dafür den richtigen Stil.

Abbildung 1: Die App 'Memo'

Das Suchfeld und die zwei Buttons sollen als Section-Header der ersten Section im TableView angezeigt werden. Fügt dazu in der XIB-Datei einen neuen *UIView* hinzu (auf der gleichen Hierarchie-Stufe wie der TableView) und macht ihn über ein Outlet mit dem dazugehörigen *UIViewController* bekannt. Dieser View soll ein *UITextField* und zwei *UIButton* enthalten. Damit der TableView diesen View als Section-Header anzeigt müssen zwei Methoden des *UITableViewDelegate* Protokolls implementiert werden. Der linke Buttons soll das System-Icon mit dem weißen Plus zeigen, er dient zum Hinzufügen neuer Memos. Der rechte Button wird eine Suche starten. Dafür gibt es kein passendes System-Icon. Nehmt dafür ein passendes Bild aus dem Internet.

1.2 Core Data

Erstellt eine neue Entity im Datenmodell der Memo-App mit dem Namen *Memo*. Fügt dieser Entity die zwei Attribute *date* und *text* hinzu und lasst Euch die Modell-Klasse von XCode erstellen. Ändert den *NSFetchedResultController* des Templates so ab, dass er mit der neuen Entity *Memo* arbeitet. Implementiert eine Action, die beim Drücken des Plus-Buttons eine neue Memo-Entity zu CoreData mit dem Text aus dem Textfeld und dem aktuellen Datum hinzufügt. Zuletzt soll die Action *Alles löschen* Buttons implementiert werden. Denkt daran, dass alle Memos aus der Datenbank einzeln gelöscht werden müssen.

1.3 Zusatzaufgabe

Beim Drücken auf den Suchen-Button dient der im Textfeld eingetragene Text als Suchbegriff. Alle Memos deren *text*-Attribut den Suchbegriff enthält sollen auf der Console ausgegeben werden. Um die Suche durchzuführen muss ein *NSFetchRequest*-Object erstellt werden. Ein *NSFetchRequest*-Objekt benötigt eine *NSEntityDescription* und ein *NSPredicate*, um die Suche durchführen zu können. Das *NSPredicate* ist die Suchanfrage und die *NSEntityDescription* gibt an was gesucht werden soll. Das für diese Aufgabe notwendige Predicate kann folgt erstellt werden:

```
[NSPredicate predicateWithFormat:@"text CONTAINS[c] %@", text]
```