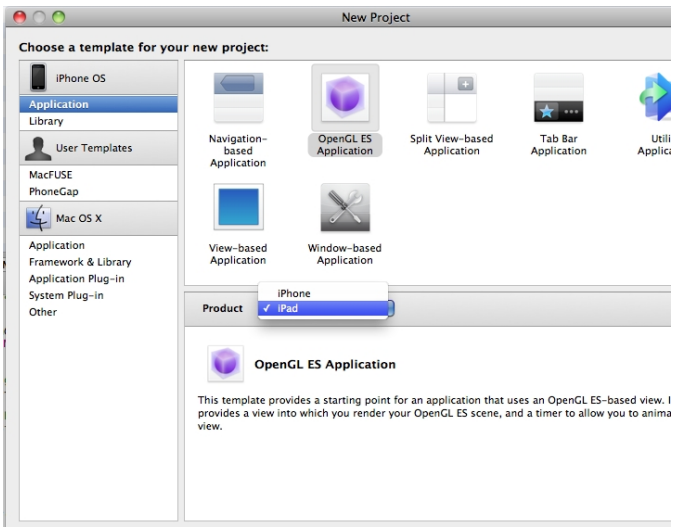
Das kürzlich erschienene iPhone OS 3.2 SDK, welches es zwar derzeit nur als beta gibt wird im folgenden näher betrachtet. Dabei wird auf Neuheiten und Unterschiede zu älteren Versionen eingegangen. Es werden nicht die einzelnen Klassen gegenüber gestellt, sondern einfach nur neue Regeln und Änderungen welche bereits bei der App-Konzeption wichtig sein könnten erläutert.

Auf den folgenden Seiten werden nun, die wie wir finden wichtigsten Punkte genauer unter die Lupe genommen ...

Installation:

Wer das neue iPhone OS SDK 3.2 installieren möchte, benötigt als erstes dafür einen Mac mit mindestens **Snow** Leopard.

Des Weiteren sollte man wissen, das sämtliche iPhone SDK Versionen 2.x genauso wie bei der Installation von 3.1.2 für Snow Leopard, deinstalliert werden. Das Programmieren für Geräte mit einer Firmware vor 3.x ist somit nicht mehr möglich. Davon betroffen sind immer noch recht viele iPod's, da die Besitzer dieser für das Firmware-Upgrade bezahlen müssen.



Xcode:

Xcode selber hat sich augenscheinlich nicht stark verändert. Der Projekt-Wizard bietet nun jedoch den neuen Applikationstypen *"Split View-Based Application"* sowie die Möglichkeit bei der Auswahl der anderen bekannten Typen, das Produkt (iPhone oder iPad) auszuwählen.

Zum iPad gibt es selbstverständlich auch einen iPad-Simulator . Dieser unterstütz zusätzlich die Schüttel-Gestik und kann ein Hardware-Keyboard simulieren. Auch läst er sich einfach per Menü auf den iPhone-Simulator umschalten.

SDK:

Alle bisherigen iPhone Klassen sind weiterhin auch im neuen SDK vorhanden. Es gab lediglich einige Änderungen an einigen Klassen. Mehr dazu findet man direkt bei Apple unter developer.apple.com/iphone/prerelease/library/releasenotes/General/iPhone32APIDiffs/index.html
.

Im neuen SDK wurden jedoch auch viele neue Klassen eingeführt. Diese sind aber laut Apple iPad Programming Guide
nicht für iPhone-Applikationen verfügbar.

Kompatibilität mit vorhandenen iPhone-Projekten:

Vorhandene iPhone-Apps lassen sich ohne weiteres auf dem iPad im sogenannten Kompatibilitätsmodus ohne irgend eine Änderung am Code ausführen. Dabei werden die iPhone-Apps in der Mitte des iPad's in Originalauflösung dargestellt. Durch das Berühren eines Vergrößerungs-Buttons kann dann die App auf den gesamten Bildschirm skaliert werden, wobei natürlich die Qualität des Aussehens darunter leidet.

Wer jedoch seine iPhone-App für iPad portieren möchte, bekommt von Apple drei geplante Möglichkeiten zur Verfügung gestellt:

- In einem Xcode Projekt eine Applikation entwickeln, welche sowohl auf iPod, iPhone als auch iPad
- In einem Xcode Projekt zwei Applikationen entwickeln, wobei die eine für iPod und iPhone und die
- für iPhone und iPad jeweils ein eigenes Xcode Projekt aufsetzen und somit zwei von einander una

Obwohl Apple selber die erste Variante empfiehlt, welche sicherlich für die meisten Entwickler auch am interessantesten erscheint, ist sie jedoch leider mit dem aktuellen Beta SDK noch nicht realisierbar.
"Creating a universal application is not supported in the initial seed release of iPhone OS 3.2. Support for creating universal binaries will be added to a later seed."

Somit muss ein Entwickler derzeit auf die 2. oder 3. Variante ausweichen oder auf die nächste SDK Version warten.

Übrigens, wer ein vorhandenes iPhone Projekt direkt für das iPad kompilieren möchte, sollte die neue Xcode Funktionalität der "Transition" nutzen. Dazu wählt der Entwickler innerhalb von Xcode das *target* des iPhone-Projektes im Projektbaum aus und wählt im Menü "*Project*" den Punkt "
Transition
". Daraufhin wird ein neues iPad-Target dem Projekt hinzugefügt und die Einstellungen dieses dementsprechend angepasst. (Ich hatte bei diesem Vorgang jedoch Probleme mit vorab selbst hinzugefügten Bibliotheken und musste unter den Target-Settings die Suchpfade für die Bibliotheken neu einstellen). Beim Kompilieren kann nun das jeweilige Target ausgewählt werden.

Neue Features:
Auf Grund der Größe des iPad's und auch der gewachsenen Anforderungen wurde das SDK natürlich um einige neue Merkmale erweitert. Eine Kurzübersicht dazu folgt nun gefolgt von genaueren Erläuterungen:

- Split-Views
- Popovers
- Konfigurierbare Modale Ansichten
- Toolbars im oberen Teil des Bildschirms
- Konfigurierbare Keyboards
- Eigene Views welche vom Keyboard Events erhalten
- Rechtschreibüberprüfung / AutoKorrektur
- Eigene Kommandos auf Editing-Menüs
- PDF - Generierung
- Gestik-Erkennung
- Konfigurierbarer und erweiterbarer Mediaplayer
- FileSharing mit angeschlossenen Desktop PC's
- Registrierung der App auf dem iPad für eigene Dateitypen

- Unterschiedliche Startbilder je nach Bildschirmausrichtung
- Anschluss von externen Bildschirmen mit einer Auflösung von bis zu 1280 x 720 Pixel

Wer besonders wichtige Änderungen lesen möchte, nutzt lieber gleich folgenden Link: Besonders wichtig für Entwickler

Split-Views:

Eines der auffälligsten neuen UI-Elemente ist die Split-View. Diese ist im Endeffekt eine vertikal zwei-geteilte View. Der linke Teil ist standardmäßig 320 Pixel breit und der rechte Teil wird auf den Rest des Bildschirms angepasst. Die Split-View soll auf dem iPad die Navigation-Bar ersetzen, bzw. empfiehlt Apple die Verwendung der Split-View im Austausch zur Navigation-Bar. Die Standardeinstellung der Split-View zeigt im Landscape-Format beide Seiten und im Portrait-Format nur die rechte Seite.

Information zur Programmierung von Split-Views findet Ihr bei Apple hier .

Popovers:

Wie der Name schon sagt, handelt es sich bei diesem UI-Element um eine nicht modale View, welche wohl überwiegend Ihren nutzen in der Darstellung von Listen bzw. Informationen zu bestimmten Objekten finden wird. Apple empfiehlt z.B. die Popover's zu nutzen, wenn eine Split-View im Portrait-Format dargestellt wird (also der linke Teil der Split-View nicht sichtbar ist) um bei Bedarf den linken Teil der Split-View in einem Popover darzustellen. **Auch wird von Apple gewünscht, Picker-Elemente, wie z.B. den Date-Picker nur noch in einem Popover zu verwenden** :

"Present a picker or date and time picker only within a popover. This placement differs from the placement recommendation for an iPhone application."

Konfigurierbare Modale Ansichten:

Auf dem iPhone wurden modale Views bisher als Vollbild-Ansicht dargestellt. Dies hat sich mit dem iPad geändert. Dort können die modalen Views, je nach Auswahl des Präsentations-Style auch kleiner als der gesamte Bildschirm sein. Dabei wird dann der restliche Teil des Bildschirms halbtransparent ausgegraut und bis auf der modalen View Touch-Events nicht mehr ausgewertet. Derzeit gibt es 4 Präsentations-Styles:

- UIModalPresentationFullScreen - Die View wird als Vollbildansicht dargestellt
- UIModalPresentationPageSheet - Die Höhe entspricht der Höhe des Bildschirms, die Breite ist jedoch kleiner
- UIModalPresentationFormSheet - Breite und Höhe sind kleiner als die Bildschirmgröße, die View wird auf dem Screen zentriert
- UIModalPresentationCurrentContext - benutzt den gleichen Style wie der Parent-ViewController

Toolbars im oberen Teil des Bildschirms:

Toolbars auf dem iPhone konnten laut Apple Style-Guide bisher nur am unteren Bildschirm dargestellt werden. Mit dem iPad hat man die Beschränkung aufgehoben und erlaubt dem

Entwickler nun die Toolbar auch am oberen Bildschirmrand zu positionieren. Apple möchte damit erreichen der Toolbar mehr Bedeutung zu zusprechen.

Konfigurierbare Keyboards:
iPhone OS 3.2 erlaubt es dem Entwickler eigene Keyboards zu implementieren. D.h. man ist nicht nur auf die vom System gegeben Keyboards angewiesen, sondern kann diese modifizieren oder selbst neue erstellen. Dies könnte dann z.B. von Vorteil sein, wenn man dem User die Möglichkeit geben möchte nur einen von z.B. zwei Werten in ein Textfeld einzutragen.

Eigene Views welche vom Keyboard Events erhalten:
Bisher gab es nur wenig UI-Elemente welche mit dem Keyboard gekoppelt waren, z.B. UITextField und UITextView. Ab iPhone OS 3.2 können solche Elemente nun selbst definiert werden. Gerade mit der Möglichkeit auch eigne Keyboards zu erstellen, entstehen ganz neue Möglichkeiten. So könnte ein Bild mit dem Keyboard gekoppelt werden, wobei auf dem Keyboard nur ein Steuerkreuz ist, welches das Bild bewegt. Mehr Infos dazu findet Ihr bei Apple hier .

Rechtschreibüberprüfung / Autokorrektur:
Dank der neuen UITextChecker Klasse unterstützt iPhone OS nun auch Rechtschreibüberprüfung bei der Texteingabe. Dabei unterstütz die Klasse für mehrere Sprachen auch das hinzufügen von neuen Wörtern oder aber von Wörtern welche ignoriert werden sollen. Apple selber beschreibt hier auch einen Weg, wie Autokorrektur realisiert werden kann. Mehr Infos zur Verwendung von UITextChecker findet ihr bei Apple hier .

Eigene Kommandos auf Editing-Menüs:
Seit dem iPhone 3G S gibt es auch die lang erwartete Funktion des Kopierens von Text und wieder einfügen an einer anderen Stelle. Um dies optisch dem Benutzer zu ermöglichen wurde das "Editig Menü" eingeführt. Dieses lässt sich nun auch neben den Systemkommandos auch mit eigenen Kommandos befüllen. Verantworlich dafür ist die Klasse UIMenuController in Kombination mit dem
UIMenuItem
. So lassen sich jetzt neben Kopieren, Ausschneiden, Einfügen, Markieren und Löschen auch selbst definierte Kommandos wie "Farbe Ändern" oder aber z.B. "Senden" realisieren. Auch die Richtung, in welche der Pfeil zeigt ist nun im
UIMenuController
einstellbar. Mehr Informationen zur Verwendung lassen sich bei Apple
hier
finden.

PDF - Generierung:
Mit dem iPhone OS 3.2 ist es erstmals möglich mit SDK-Klassen eigene PDF-Dateien zu generieren. Dabei werden die die PDF-Dateien einfach mit dem UIKit erstellt, nachdem erstmal

ein PDF-Canvas festgelegt wurde. Ebenso sind Links in den PDF-Dateien möglich. Mehr Informationen, wie eine PDF-Datei erstellt wird, findet Ihr hier .

Gestik-Erkennung:

Eine weitere Erneuerung in iPhone OS 3.2 ist die Erkennung von speziellen Gestiken. Bisher musste der Entwickler selbst versuchen zu erkennen, wenn der Anwender beispielsweise einen Zoom oder eine Drehung der Views erzwingen wollte. Dies wird nun weitestgehend vom SDK erkannt. Dabei unterscheidet das SDK in folgende Gestiken:

- einfache Touches (UITapGestureRecognizer) - einzelne Berührungen auf dem Bildschirm werden erkannt und ggf. auch durch mehrere Finger

- Zooming (UIPinchGestureRecognizer) - das übliche Zooming wird erkannt, d.h. wenn zwei Finger sich aufeinander zu oder auch weg bewegen

- Verschiebung (UIPanGestureRecognizer) - ein oder mehr Finger werden erkannt wenn ein Objekt verschoben werden soll
- Ziehen (UISwipeGestureRecognizer) - Ein Ziehen wird erkannt, wenn ein oder mehr Finger sich in einer vorher definierte Richtung in eine bestimmte Reichweite bewegen

- Rotation (UIRotationGestureRecognizer) - Zwei Finger bewegen sich in entgegengesetzte Kreisbewegungen um eine Rotation hervor zu rufen

- Lange Berührung (UILongPressGestureRecognizer) - wird erkannt wenn der Anwender einen oder mehr Finger für eine vorher definierte Zeit zwar berührt aber nicht bewegt.

Mehr Informationen zur Implementierung von Gestiken gibt es hier .

Konfigurierbarer und erweiterbarer Mediaplayer:

Wer bisher viel Dynamik bei Verwendung des Mediaplayer erwartet hatte, wurde entäuscht. So wurde bisher der Mediaplayer immer im Vollbildmodus ausgeführt und hatte keinerlei Möglichkeiten geboten, das User-Interface der Bedienungsknöpfe zu ändern. Dies hat sich nun mit iPhone OS 3.2 geändert. So ist es von nun an möglich Start- und Endpunkte eines Videos vorab zu bestimmen (im aktuellen BETA SDK nur der Startpunkt). Auch die Erstellung einer Vorabansicht (Thumbnail) eines Videos ist möglich. Aber die wohl entscheidensten Änderungen sind, das sich einerseits ein Video innerhalb einer View abspielen läst (also ohne Vollbildmodus) oder sogar im Hintergrund und das sich nun auch eigene Elemente zur Steuerung des Videos hinzufügen lassen, bzw lässt sich das Video mit eigenen Views überlagern. Auch kann die Hintergrundansicht eines Videos beliebig angepasst werden.

FileSharing mit angeschlossenen Desktop PC's:

Weiterhin neu und wahrscheinlich für viele iPhone OS Entwickler revolutionär ist, das sich eigene vom Programm ersteltte Dateien mit einem angeschlossenen Desktop-PC austauschen lassen. Das bedeutet, das Dateien, welche mit einer iPad-Applikatiom erstellt werden nun auch auf dem heimischen Desktop-PC gelangen können um dort zur eventuellen Weiterverarbeitung

genutzt zu werden und umgekehrt. Der iPad ist dabei sogar in der Lage, das Verzeichnis des PC zum Datenaustausch anzuzeigen. Dateien, welche für den Austausch genutzt werden sollen, müssen auf dem iPad im Programmorder unterhalb von Documents/Shared liegen.

Leider lassen sich diese Dateien jedoch nicht mit anderen iPad-Applikationen wieder verwenden, da weiterhin alle Programme auf dem iPad in einer eigenen Sandbox betrieben werden. Dennoch, lässt man seine Phantasie etwas schweifen, wären z.B. verschiedenste Text-Editoren möglich, mit welchen man Dateien, welche vorab auf dem Desktop-PC editiert wurden nun auch auf dem iPad bearbeitet werden können.

Registrierung der App auf dem iPad für eigene Dateitypen:

iPad-Programme lassen sich nun für beliebige Dateitypen regiestrieren, so das bei Aufruf eines bestimmten Dateitypen, eine iPad-Applikation automatisch gestartet werden kann. Um diese Funktionalität zu unterstützen hilf der UIDocumentInteractionController weiter.

Unterschiedliche Startbilder je nach Bildschirmausrichtung:

Wer bisher auf dem iPhone eine Applikation geöffnet hat, konnte das Startbild nur in einer Orietierungsausrichtung sehen. Mit iPhone OS 3.2 hat sich dies geändert. Der Entwickler hat von nun an die Möglichkeit für das Querformat oder Portraitfomrat jeweils ein Startbild festzulegen. Die Einstellungen dafür werden in der info.plist gehandhabt.

Anschluss von externen Bildschirmen mit einer Auflösung von bis zu 1280 x 720 Pixel:

Das iPad unterstützt den Anschluss von externen Anzeigegeräten mit einer Auflösung von bis zu 1280x720 Pixeln. Alles was darüber hinausgeht wird hoch skaliert. Dabei soll die Übertragung auch für "normale" Apps funktionieren, im Gegensatz zum iPhone, wo nur spezielle Apps wie der Mediaplayer die Screendaten auf ein externen Gerät übertragen konnte.

"Applications can use this connection to present content in addition to the content on the device’s main screen."

Besonders wichtig für Entwickler:

So wie man das aus den Apple-Dokumenten raus liest, ist eine der entscheidensten Änderungen zum iPhone, das eine iPad-Applikation immer sämtliche Bildschirmausrichtungen unterstüzen muß. Dazu hier zwei Zitate aus den Guidlines:

"An application’s interface should support all landscape and portrait orientations. This behavior differs slightly from the iPhone, where running in both portrait and landscape modes is not required."

"Important: Although you might not have supported all orientations in your iPhone application, you must do so in your iPad application."

Eine weitere neue Änderung, ist, dass das Application-Icon nun nicht mehr in einer Auflösung von 54x54 Pixeln erstellt werden soll, sondern in 72x72 Pixeln. Dies resultiert sicherlich aus der großen Bildschirmdiagonale des iPad's.

Resümee:

Bleibt als einizigstes zu überlegen, wie Apple es schaffen will, das in einem Xcode-Projekt mit nur einem *target*, es geschafft werden soll, dass die resultierende Binärdatei für das iPhone und den iPad funktionieren soll. Gerade dann, wenn Apple schreibt, das alle zu iPhone OS 3.2 hinzugefügten Klassen nur auf dem iPad funktionieren:

"All of the classes available for use in iPhone applications are also available in iPad applications. (Classes introduced in iPhone OS 3.2 are not available for use in iPhone applications.)"

Quellen:

http://developer.apple.com/iphone/prerelease/library/documentation/General/Conceptual/iPadProgrammingGuide

http://developer.apple.com/iphone/prerelease/library/documentation/General/Conceptual/iPadHIG/UIElements/UIElements.html