

Mac- Programmierung

FÜR **KIDS**



Inklusive CD-ROM

Holger Hinzberg



Mac- Programmierung für Kids

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <<http://dnb.d-nb.de>> abrufbar.

Bei der Herstellung des Werkes haben wir uns zukunftsbewusst für umweltverträgliche und wiederverwertbare Materialien entschieden. Der Inhalt ist auf elementar chlorfreiem Papier gedruckt.

ISBN: 978-3-8266-8685-6

2. Auflage 2012

www.mitp.de

E-Mail: kundenbetreuung@hjr-verlag.de

Telefon: +49 6221 / 489 -555

Telefax: +49 6221 / 489 -410

© 2012 mitp, eine Marke der Verlagsgruppe Hüthig Jehle Rehm GmbH
Heidelberg, München, Landsberg, Frechen, Hamburg.

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Sabine Schulz

Sprachkorrektorat: Manfred Buchholz

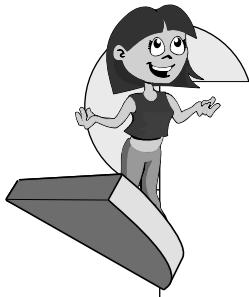
Covergestaltung: Christian Kalkert, www.kalkert.de

Satz: Johann-Christian Hanke



Inhaltsverzeichnis

	Vorwort	15
	Einleitung	16
	Der Aufbau dieses Buchs	16
	Wie arbeitest du mit diesem Buch?	17
	Was brauchst du für dieses Buch?	18
	Die Geschichte vom Code und vom Compiler	19
	Ohne Programmiersprache geht es nicht	20
	Die Sache mit der Tastatur	22
	Die Sache mit der Maus	22
1	Xcode und dein erstes Projekt	25
	Die Installation von Xcode	26
	Eine Projektvorlage auswählen	28
	Das Workspace-Fenster	30
	Jetzt wird programmiert!	33
	Code Sense – Die Eingabehilfe	36
	Auch Kommentare sind wichtig	37
	Ein Framework wird importiert	39
	Die Methode main	40
	Am Ende folgt return	41
	Nützliche Einstellungen für Xcode	42
	Änderungen am Workspace-Fenster	45
	Schnellhilfe und Jump Bar	46
	Zusammenfassung	48
	Ein paar Fragen ...	48
2	Variablen und Datentypen	49
	Was sind Variablen?	50
	Ein neues Projekt für Variablen	52



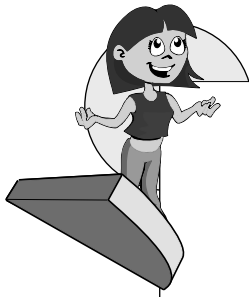
Man kann sogar rechnen!	55
Plus Plus	57
Manche Variablen sind konstant	58
Symbole ersetzen mit #define	60
Es könnte so einfach sein	61
Mathematik	62
Zeiger	64
Zusammenfassung	65
Ein paar Fragen ...	66
... und zwei Aufgaben	66

3 Kontrollstrukturen 67

Fallunterscheidungen und Schleifen	68
Die if-Struktur	69
Vergleichen von zwei Zahlen	71
Die if-Struktur im Detail	73
switch und case	74
Die for-Schleife	77
Der Geltungsbereich von Laufvariablen	79
Mit double und rückwärts	81
Ein Zinsrechner mit for-Schleife	81
Ein Zinsrechner mit while-Schleife	83
Die do-while-Schleife	85
Schleifen abbrechen mit break	85
Einen Durchlauf abbrechen mit continue	88
Erweiterte Vergleichsausdrücke	89
Endlosschleifen	90
Zusammenfassung	94
Ein paar Fragen ...	94
... und eine Aufgabe	94



4	Funktionen	95
	Ein Zinsrechner mit Funktionen	96
	Die Parameter für die Funktion	97
	Mit Rückgabewert ist es besser	102
	Funktionsdeklarationen	105
	Mehr Funktionen für mehr Funktionalität	108
	Die Übergabe als Wert	111
	Berechnungen am Kreis	114
	Mehr Übersicht mit Code-Faltung	116
	Zusammenfassung	117
	Ein paar Fragen ...	118
	... und eine Aufgabe	118
5	Klassen und Objekte	119
	Ein Rückblick in vergangene Zeiten	120
	Ein Zinsrechner mit Klassen	121
	Die zwei Dateien einer Klasse	125
	Auch Methoden sind nicht schwer	126
	Klassen sind nur Baupläne	128
	Objekte brauchen Speicherplatz	129
	Nachrichten senden	132
	Vererbung	134
	Speicherverwaltung	136
	Der Garbage Collector	138
	Zusammenfassung	141
	Ein paar Fragen ...	141
	... und eine Aufgabe	141
6	Klassen als Datenspeicher	143
	Accessor-Methoden	144
	Getter und Setter für Zeichenketten	149
	Deklarierte Eigenschaften	153



Punkt-Notation	155
Zuweisungen über eigenen Methoden	156
Initialisierungs-Methoden	159
Eine gute Beschreibung	160
Objekte vergleichen	162
Klassen als Controller	167
Model-View-Controller	168
Zusammenfassung	170
Ein paar Fragen ...	170
... und eine Aufgabe	170

7 Hallo Welt – eine Cocoa-Anwendung 171

Das Cocoa-Framework	172
Ein Controller muss sein	176
Ohne Action und Outlet geht nichts	177
Der Interface Builder	180
Der Inspector weiß alles!	184
Eine Instanz für den Controller	188
Die richtigen Verbindungen sind wichtig	189
Zusammenfassung	192
Ein paar Fragen ...	192
... und eine Aufgabe	193

8 Cocoa mit Sahne 195

Dock und Outline View im Interface Builder	196
Der Inspector unter der Lupe	198
Die Größenanpassung von Steuerelementen	203
Mehr Action für Hallo Welt	206
Eine Anwendung erwacht	212
Delegation für den Controller	214
Wo ist meine Anwendung?	216
Zusammenfassung	219



Ein paar Fragen ...	219
... und eine Aufgabe	219

9 Der Zinsrechner 221

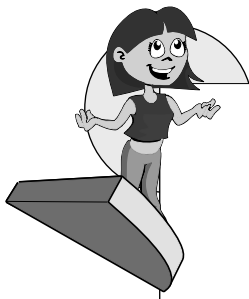
Eine grafische Oberfläche	222
Controller und Assistant	224
Code für den Controller	230
Nicht immer alles neu programmieren	231
Anpassen der Textausrichtung	235
Zahlen formatieren	235
Eigene Zahlenformate erstellen	240
Das EVA-Prinzip	242
Zusammenfassung	242
Ein paar Fragen ...	243
... und eine Aufgabe	243

10 Ein Schnappschuss – bitte lächeln! 245

Einen Schnappschuss anlegen	246
Ein Objekt für alle	248
Schnappschüsse wiederherstellen	251
Ein neuer Name für den Controller	253
Die Tabulatorreihenfolge	254
Der initialFirstResponder	257
Zusammenfassung	259
Ein paar Fragen ...	259
... und eine Aufgabe	260

11 Lotto – dem Zufall auf der Spur 261

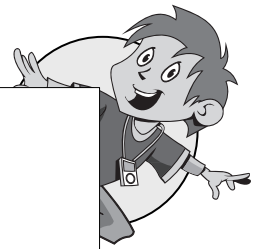
Der Zufallszahlengenerator	263
Ein Fenster für die Lottozahlen	266
Zahlen verwalten mit einem Array	268
Jetzt wird es kompliziert	272
Der Icon Composer	275



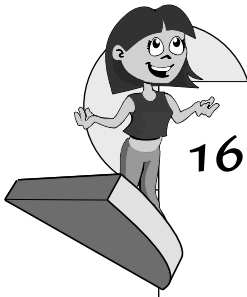
Ein Icon für das Projekt	279
Zusammenfassung	281
Ein paar Fragen ...	281
... und drei Aufgaben	282

12	Eine Kiste voller Steuerelemente	283
	Ein Umrechner für Temperaturen	284
	Von Celsius nach Fahrenheit und zurück	288
	Ein digitaler Einkaufszettel	292
	Eine Checkbox ist auch nur ein Button	292
	Text mit Umbruch	294
	Von der Checkbox zum Text	297
	Zeichenketten beschneiden mit NSRange	301
	Schieberegler und Fortschrittsbalken	304
	Ein wenig Formatierung schadet nie	309
	Zusammenfassung	311
	Ein paar Fragen ...	311
	... und eine Aufgabe	311

13	Fehlersuche mit dem Debugger	313
	Ein letztes Command Line Tool-Projekt	314
	Haltepunkte	316
	Informationen aus dem Datatip	317
	Die Debug-Leiste	318
	Den Programmablauf verfolgen	319
	Die Variablenansicht	322
	Der Callstack	323
	Die Manipulation von Variablen	324
	Lottozahlen Debugging	326
	Zusammenfassung	329
	Ein paar Fragen ...	329
	... und eine Aufgabe	329



14	Eine Mini-Datenbank	331
	Objekte bitte in das Array einsteigen	334
	Objekte bitte wieder aussteigen	337
	Es geht vor und zurück	340
	Nicht über die Grenze treten!	341
	Eine ganz andere Lösung	343
	Ein Blick auf den Callstack	346
	Das Info-Fenster	347
	Zusammenfassung	351
	Ein paar Fragen ...	351
	... und eine Aufgabe	352
15	MiniDB – Eingaben speichern und laden	353
	Neue Aufgaben für neue Methoden	355
	Von der Oberfläche in das Array	357
	Abbrechen – das ist einfach!	359
	Ein Alarmfenster	359
	Piep, Piep, NSBeep	360
	Und was ist mit Abspeichern?	361
	Alles eine Frage des richtigen Protokolls	362
	Jetzt wird alles codiert	365
	Laden und Speichern sind auch nur Methoden	366
	Das NSSavePanel	367
	Dateien archivieren mit dem NSKeyedArchiver	369
	Dateien laden mit dem NSKeyedUnarchiver	372
	Einblicke mit dem Debugger	375
	Aufräumarbeiten am Array	376
	Zusammenfassung	377
	Ein paar Fragen ...	378
	... und eine Aufgabe	378



16

MiniDB – Menü- und Symbolleiste 379

- Es kann aufgeräumt werden 384
- Ein Menu Item ist kein Button 385
- Refactoring für die Outlets 387
- Den Programmnamen ändern 388
- Eine Anwendung ist nur ein Bündel 391
- Die Symbolleiste 392
- Ein Einsatz für den Icon Composer 397
- Symbole für die Symbolleiste 399
- Übersetzen der Symbolleiste 403
- Zusammenfassung 405
- Ein paar Fragen ... 406
- ... und eine Aufgabe 406

17

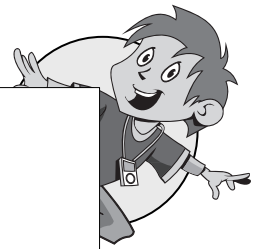
Timer – Die Zeit läuft! 407

- Die Klasse NSTimer 408
- Was tickt denn da? 410
- Die Stoppuhr 411
- Große Schrift? Kleines Problem! 413
- Hallo Empfänger – hier Sender! 416
- Die Zeit vergeht so schnell ... 418
- Eine ganz besondere Uhr 420
- Uhrzeit aus Fortschrittsbalken 424
- 3 – 2 – 1 – Countdown 428
- Zusammenfassung 432
- Ein paar Fragen ... 432
- ... und eine Aufgabe 432

18

Zeit zum Spielen 433

- Schritt für Schritt zur fertigen Oberfläche 434
- Weiter geht es im Programmcode 436
- Neues Spiel – Neues Glück 439



Raten bedeutet vergleichen	441
Einige kleine Verbesserungen	442
Spiel auf Zeit	447
Noch ein kleiner Fehler	451
Schlag den Maulwurf	453
Jeder Maulwurf ist ein Outlet	455
Zwei Timer für ein Spiel	457
Er ist da – Er ist weg	461
Ein Button ist kein Maulwurf	464
Zusammenfassung	465
Ein paar Fragen ...	465
... und drei Aufgaben	466

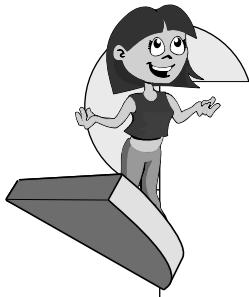
19	MiniDB – Sheet und Badge	467
	Leichte Anpassungen der Oberfläche	468
	Fenster und ihre Outlets	469
	Ein Sheet öffnen ist nicht schwer ...	474
	... ein Sheet schließen auch nicht	475
	Der Anfang bestimmt das Ende	477
	Sheet geschlossen – Die Arbeit beginnt	478
	Letzte Aufräumarbeiten	480
	Klein aber fein – Ein Badge für das Icon	482
	Zusammenfassung	485
	Ein paar Fragen ...	485
	... und eine Aufgabe	485

Anhang A 487

Für Eltern ...	487
... und für Lehrer	487

Anhang B 489

Warum heißt alles NS-irgendwas?	489
---------------------------------	-----



Anhang C 491

Fragen und Antworten 491

Danksagung 500

Stichwortverzeichnis 501



Vorwort

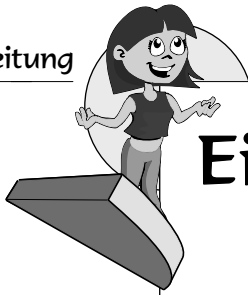
Wenn du dieses Buch gekauft hast oder es vielleicht gerade kaufen willst, hat dich möglicherweise auch die Leidenschaft des Programmierens gepackt. Eventuell bist du aber auch nur neugierig und möchtest wissen, was du mit deinem Computer alles machen kannst, wenn du nicht nur fertige Programme von anderen benutzt.

Zu behaupten, das Programmieren auf dem Mac zu erlernen wäre leicht, ist sicher nicht die ganze Wahrheit. Genau wie bei einer Fremdsprache, die man lernen will, ist auch hier aller Anfang schwer. Aber je mehr man weiß, umso leichter geht es voran – und umso mehr Spaß macht es.

Dieses Buch und die Beispiele darin sollen auch Spaß machen. Zwar wird dir vieles neu und unbekannt vorkommen, besonders dann, wenn du noch nie in deinem Leben programmiert hast, aber mit vielen kleinen Schritten wirst du immer mehr lernen und immer mehr erfahren. Doch selbst wenn du irgendwann jahrelang deine eigenen Programme geschrieben hast, wirst du immer noch etwas Neues lernen können. Das ist das Tolle am Programmieren, aber auch die Herausforderung.

Mit deinem Mac und dem Betriebssystem OS X steht dir eine Plattform zur Verfügung, auf der du viele eindrucksvolle Dinge schaffen kannst. Dinge, von denen Softwareentwickler vor wenigen Jahren noch geträumt haben. Da dir Apple alle Werkzeuge, die du zur Entwicklung eigener Programme brauchst, kostenlos zur Verfügung stellt, gibt es keinen Grund, warum nicht auch du programmieren solltest.

Nur keine Angst. Es macht Spaß!



Einleitung

Der Aufbau dieses Buchs

Die ersten Beispiele in diesem Buch geben eine kurze Einführung in das Programm Xcode und in die Programmiersprache Objective-C. Xcode ist die Entwicklungsumgebung für das OS X Betriebssystem und somit für dich das wichtigste Werkzeug, wenn du auf dem Mac programmieren möchtest. Mit Xcode werden deine Eingaben in eigenständige, lauffähige Programme umgewandelt. Objective-C ist die Programmiersprache, die in Xcode verwendet wird, und auch die Entwickler von Apple verwenden Objective-C für ihre Programme. Wenn du schon Erfahrung mit anderen Programmiersprachen wie zum Beispiel Visual Basic oder Java hast, wird dir manches bekannt vorkommen. Andere Dinge sind hingegen typisch für Objective-C und man findet sie in anderen Sprachen eher selten oder gar nicht.

Der überwiegende Teil dieses Buches beschäftigt sich aber mit der Programmierung von grafischen Benutzeroberflächen für Programme, mit Fenstern, Schaltflächen und vielen anderen Dinge, die dazugehören. Grundlage dieser Programme ist das sogenannte »Application Kit Framework«, eine von Apple zur Verfügung gestellte Sammlung von Bausteinen für grafische Anwendungen.

Besondere Vorkenntnisse für dieses Buch brauchst du nicht. Wenn du deinen Mac gut bedienen kannst, dir auch die Systemeinstellungen nicht unbekannt sind, sollte es keine Probleme geben. Die ersten Beispiele fangen leicht an und die Programme sind so einfach, dass sie noch in einem einzelnen Kapitel behandelt werden können. Später, wenn die Programme komplizierter werden, wird ein Kapitel nicht mehr ausreichen und eine Anwendung wird dich auch über einen längeren Zeitraum begleiten. Du wirst lernen, wie du deine Programme immer weiter verbesserst und verfeinerst, indem du ein Menü oder eine Symbolleiste hinzufügst oder dem Programm ein eigenes Symbol im Dock zuweist, damit es auch dort einen guten Eindruck macht.



Wie arbeitest du mit diesem Buch?

Grundsätzlich besteht dieses Buch aus einer Menge Text mit vielen Abbildungen dazwischen. Natürlich habe ich mich bemüht, alles so aufzubereiten, dass daraus lauter leicht verdauliche Happen werden. Damit das Ganze noch genießbarer wird, gibt es zusätzlich noch einige Symbole, die ich dir hier gerne erklären möchte:

Arbeitsschritte

➤ Wenn du dieses Zeichen siehst, heißt das: Es gibt etwas zu tun. Damit kommen wir beim Programmieren Schritt für Schritt einem neuen Ziel immer näher.

Grundsätzlich lernt man besser, wenn man einen Programmtext selbst eingibt oder ändert. Aber nicht immer hat man große Lust dazu. Weil alle Projekte im Buch auch auf der CD sind, findest du dort für jedes Kapitel einen Ordner mit den Beispielen. Wenn du also das Projekt nicht selbst erstellen willst, kannst du stattdessen die zugehörige Datei laden.

Aufgaben

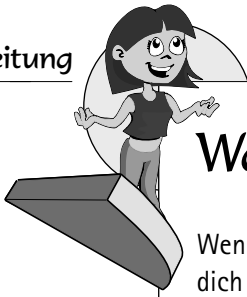
Am Ende der meisten Kapitel folgen ein paar Fragen, mit denen du kontrollieren kannst, ob du alles verstanden hast. Dann gibt es Aufgaben und kleine Übungen, um deine Programmierkenntnisse zu fordern. Die Antworten zu den Fragen findest du im Anhang des Buches und die Aufgaben als Projekte auf der CD. Aber auch zwischendurch gibt es immer mal wieder Vorschläge, was du mit dem neu erworbenen Wissen noch alles programmieren kannst.

Wichtige Stellen im Buch

Hin und wieder findest du ein solch dickes Ausrufezeichen im Buch. Das ist dann eine Stelle, an der etwas besonders Wichtiges steht.

Wenn es um eine ausführlichere Erläuterung geht, tritt Buffi in Erscheinung und bringt dir was aus seiner Kiste mit Tipps & Tricks.





Was brauchst du für dieses Buch?

Wenn du einen neuen oder nicht zu alten Mac hast, gibt es nichts, was dich vom Programmieren abhalten könnte, da du die meisten Dinge, die du benötigst, schon hast oder kostenlos bekommen kannst.

Betriebssystem

Um mit diesem Buch das Programmieren auf dem Mac zu erlernen, benötigst du auf deinem Computer eine aktuelle Version des Apple Betriebssystems OS X, jedoch mindestens die Version 10.7 »Lion«. Was für einen Computer du genau hast, spielt keine Rolle. Ein iMac ist genauso gut geeignet wie ein MacBook oder ein MacMini. Du kannst auf jedem dieser Rechner Programme für alle anderen erstellen. Das war nicht immer so. Um für den ersten Macintosh-Computer zu programmieren, benötigte man einen ganz anderen Rechner, die Lisa, ebenfalls von der Firma Apple.

Programme

Programmiert wird auf dem Mac mit dem Programm Xcode. Xcode ist eine Entwicklungsumgebung auch IDE (Integrated Development Environment) genannt. Obwohl Xcode zweifellos das wichtigste Programm für einen Entwickler ist, ist es doch nur Teil eines Paketes, das sich »Xcode Developer Tools«, übersetzt »Werkzeuge für Entwickler«, nennt. Dieses Paket enthält neben dem Programm Xcode noch weitere nützliche Anwendungen, von denen du einige in diesem Buch kennenlernen wirst.

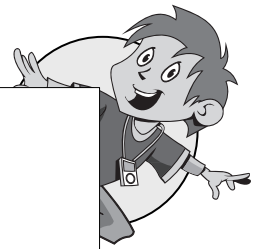
Xcode mit den Werkzeugen für Entwickler bekommst du kostenlos im Mac App Store und kannst es von dort auf deinen Computer herunterladen.

Im Internet

Auf meiner Internetseite zum Buch findest du außer den Beispielen zum Herunterladen auch Informationen zu anderen Themen, die es zwar nicht in das Buch geschafft haben, die dich aber vielleicht auch interessieren.

<http://www.cocoa-coding.de/kids/>

Sollten sich Fehler im Buch finden oder sollte Apple Anpassungen an Xcode vornehmen, werde ich versuchen, diese auf meiner Seite zu protokollieren, damit du auch weiterhin problemlos mit dem Buch arbeiten kannst. Bei Fragen zu den Beispielen oder wenn du selbst Fehler findest, kannst du dich gerne auch direkt an mich wenden, meine E-Mail-Adresse findest du ebenfalls auf der Internetseite.



Die Geschichte vom Code und vom Compiler

Du hast vielleicht schon mal den Satz gehört, dass ein Computer nur 0 und 1 versteht. In Wirklichkeit ist es noch viel schlimmer, denn da ein Computer mit elektrischem Strom arbeitet, kann er nur zwischen den Zuständen »Strom fließt« und »Strom fließt nicht« unterscheiden. Kombinationen aus diesen beiden Möglichkeiten veranlassen den Rechner dann, etwas zu tun. Zum Beispiel zwei Zahlen zu addieren oder einen Text auf den Bildschirm zu schreiben.

Natürlich wäre es sehr mühsam einen Computer zu programmieren, indem man durch verschiedene Schalter elektrischen Strom zum Fließen bringt. Die ersten Computer funktionierten tatsächlich noch so, aber diese Zeiten sind glücklicherweise lange vorbei. An der grundlegenden Funktion der Computer hat sich allerdings nichts geändert. Noch immer muss elektrischer Strom gesteuert werden. Allerdings ist die Eingabe der Programme heute etwas komfortabler. Ein Programm wird nicht mehr durch verschiedene Schalter erzeugt, sondern durch Anweisungen in einer Programmiersprache. Für so eine Auflistung von Befehlen gibt es viele Namen. Man nennt sie Quelltext, Programm(code), im Englischen »Sourcecode«, oder manchmal auch nur Code.

Wie auch in einer normalen Fremdsprache setzt sich der Programmcode aus »Vokabeln« und »Grammatik« zusammen. Wenn man eine Anweisung falsch schreibt, wird sie der Computer nicht verstehen, und wenn man Anweisungen in der falschen Reihenfolge verwendet, wird nicht das passieren, was man möchte. Der Vergleich mit einer Fremdsprache geht sogar noch weiter, denn je mehr Vokabeln du kennst, umso mehr kannst du in der Sprache erreichen, und umso leichter ist der Umgang mit dieser Sprache.

Leider ist der Code, auch wenn er in einer Programmiersprache geschrieben wurde, nur ein Text, mit dem ein Computer zunächst nicht viel anfangen kann. Das Programm muss wieder in eine Serie von 0 und 1, also in Signale von elektrischem Strom, umgewandelt werden. Diese Aufgabe übernimmt ein Programm, das Compiler heißt. Dieser Compiler, zu Deutsch »Übersetzer«, wandelt den Programmcode in ein für den Computer geeignetes Format um, das man auch Maschinensprache nennt. Gleichzeitig entdeckt der Compiler bei der Übersetzung Fehler, wenn die verwendeten Anweisungen nicht denen entsprechen, die der Compiler kennt.



Ein weiteres Programm, mit dem man als Entwickler aber wenig zu tun hat, ist der Linker. Der Linker, zu Deutsch »Verknüpfer« oder »Binder«, hat die Aufgabe, die vom Compiler übersetzten Einzelteile zu einem großen Ganzen zusammenzusetzen. Das passiert in der Regel automatisch, ohne dass man selbst etwas machen muss.

In der Vergangenheit waren Compiler, Linker und der Texteditor für den Programmcode oft eigenständige Programme, die vom Entwickler einzeln bedient werden mussten. Moderne Werkzeuge wie Xcode fassen aber alle Funktionen zusammen und tragen daher die Bezeichnung »integrierte Entwicklungsumgebung«.

Unabhängig davon, wie schnell dein Rechner sein mag, wird die Arbeit von Compiler und Linker immer einige Zeit in Anspruch nehmen. Glücklicherweise nicht mehr einige Stunden, wie es in den frühen Jahren der Datenverarbeitung nicht ungewöhnlich war. Bei einem umfangreichen Programm kann das zwar durchaus einige Minuten dauern – bei den Beispielen in diesem Buch jedoch nur wenige Sekunden.

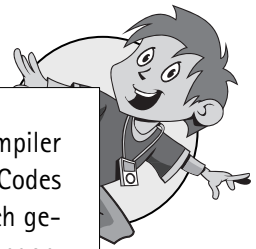
Ohne Programmiersprache geht es nicht

Falls du neugierig darauf bist, was für Programme du vielleicht in Zukunft schreiben kannst, dann sieh dir mal die Software an, die mit OS X ausgeliefert wurde. Auch iTunes und iPhoto wurden bei Apple in Objective-C erstellt und viele der dort verwendeten Bauteile dieser Programme stehen dir auch für eigene Anwendungen zur Verfügung.

Objective-C ist eine objektorientierte Programmiersprache. Was das genau bedeutet, wirst du in späteren Kapiteln noch genauer erfahren. Eines der Hauptmerkmale einer solchen Sprache ist die hohe Wiederverwendbarkeit von Programmteilen. Wenn du also einen Programmteil, wie zum Beispiel eine komplizierte Berechnung, erfolgreich programmiert hast, kannst du ihn später sehr leicht in anderen Programmen wiederverwenden.

Bevor du allerdings anfangen kannst zu programmieren, solltest du zumindest einige der wichtigsten Grammatikregeln dieser Programmiersprache kennen.

In Objective-C folgt am Ende jeder Anweisung ein Semikolon (;). Dieses Zeichen signalisiert dem Compiler eindeutig, dass der Befehl dort zu Ende ist und dass die folgende Anweisung ein neuer Befehl sein soll. Das ist nötig, da nicht alle Anweisungen gleich lang und gleich aufgebaut sind. Setzt



man aber ein Semikolon zu früh oder vergisst man es, wird der Compiler das in den meisten Fällen bemerken und bei der Übersetzung des Codes bemängeln. Man kann allerdings auch Pech haben und durch ein falsch gesetztes Semikolon entstehen andere, aber trotzdem gültige, Anweisungen. Das ist einer der Gründe, warum der Compiler Fehler im Programmcode zwar erkennt, aber nicht selbstständig korrigiert.

Während des Programmierens gibt es immer wieder Dinge, die für den Menschen und nicht für den Computer wichtig sind. Die Formatierung und das Layout des Programmcodes sind das beste Beispiel. So ist es möglich, mehrere Anweisungen einfach hintereinander zu schreiben. Natürlich immer mit einem Semikolon getrennt.

```
Anweisung1;Anweisung2;Anweisung3;
```

Das würde problemlos funktionieren, besser ist es aber, jede Anweisung in eine neue Zeile zu schreiben.

```
Anweisung1;  
Anweisung2;  
Anweisung3;
```

Die Formatierung des Codes ist für den Compiler natürlich vollkommen uninteressant. Am Semikolon erkennt er genau, wann eine Anweisung zu Ende ist. Ob die Befehle in einer Zeile hintereinander oder in mehreren Zeilen untereinander stehen, macht also für ihn überhaupt keinen Unterschied. Aber auch der Programmierer sollte in der Lage sein, das Programm zu lesen und seine Funktion nachvollziehen zu können. Das geht viel besser, wenn der Programmcode gut formatiert ist. Später wirst du Anweisungen kennenlernen, die ineinander verschachtelt sind, und dann ist leicht lesbarer Code wirklich sehr wichtig. Es gibt nämlich auch Befehle, die aus mehr als einer Zeile bestehen.

```
Anweisung;  
{  
    Anweisung;  
}
```

So ein Konstrukt aus geschweiften Klammern nennt man Block und man benutzt Blöcke immer dann, wenn man Befehle mehrfach oder nur unter bestimmten Bedingungen ausführen will. Auch hier sollte man den Code gut formatieren und die Anweisungen im Block einrücken. Die Tabulator-taste ist dabei eine große Hilfe. Es gibt aber auch Einstellungen in Xcode, die dich bei der Formatierung des Codes unterstützen. Außerdem kannst du Kommentare in deinen Quelltext schreiben, um dir direkt bei den Anweisungen kleine Notizen zu machen.



Das alles klingt für dich vielleicht sehr theoretisch und es wurde hier auch schon einiges vorweggenommen, was später noch sehr detailliert erklärt wird. Lass dich davon am besten nicht beeindrucken. Wenn du die ersten Programme selbst geschrieben hast, wirst du bestimmt vieles besser verstehen. Lies diese Einleitung einfach später noch einmal und alles wird dir ganz leicht und verständlich vorkommen. Programmieren lernt man nicht in der Theorie, sondern nur, indem man wirklich Programme schreibt. Deshalb geht es im nächsten Kapitel auch sofort richtig los. Dein erstes eigenes Projekt wartet!

Die Sache mit der Tastatur

Manche der Zeichen, die man zum Programmieren ständig benötigt, sind leider auf der deutschen Tastatur nicht abgebildet. Man kann sie zwar problemlos eingeben, oft ist aber etwas Eingewöhnung nötig, bis man sich gemerkt hat, wo diese Zeichen verborgen sind. Hauptsächlich handelt es sich dabei um Klammern.

Die eckigen Klammern [] erreichst du mit der **Alt**-Taste und den Zahlentasten **5** und **6**.

Die geschweiften Klammern { } erreichst du mit der **Alt**-Taste und den Zahlentasten **8** und **9**.

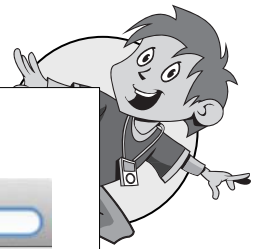
Das funktioniert aber nur mit den Zahlentasten auf dem Haupttastenblock. Mit den Zahlentasten des Nummernblocks funktioniert es nicht.

Die Sache mit der Maus

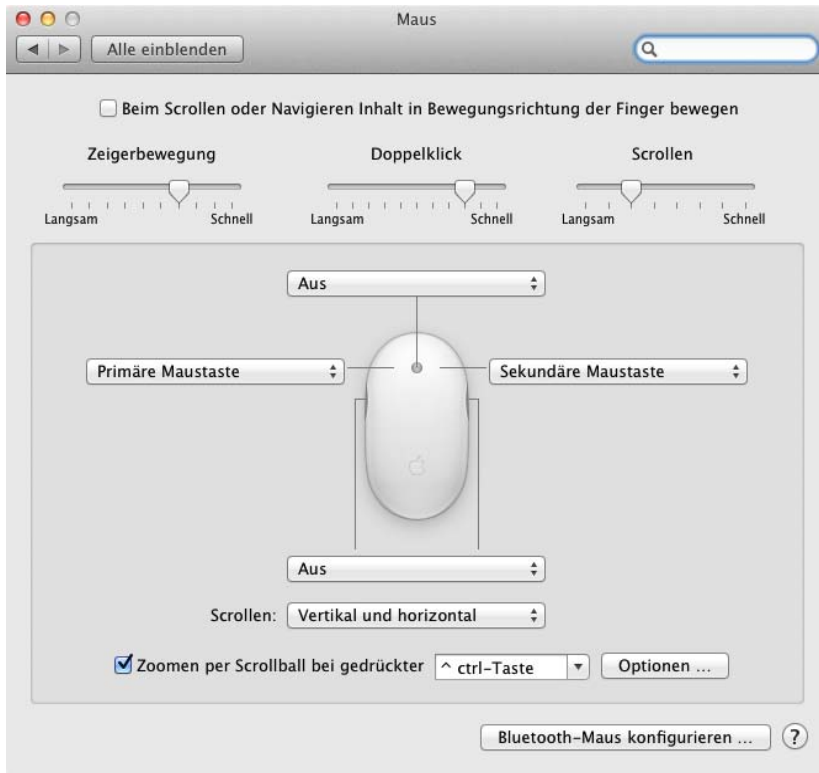
In der Vergangenheit hatten Mäuse der Mac Computer manchmal nur eine Taste, und obwohl die Apple Mouse und auch die Magic Mouse inzwischen sogar mehr als zwei Tasten haben, muss man die zusätzlichen Funktionen erst selbst konfigurieren.

In Xcode, dem Programmierwerkzeug auf dem Mac, gibt es sehr viele Befehle, die man durch einen Klick auf die sekundäre Maustaste ausführen kann. Deshalb ist es sinnvoll, die Maus so anzupassen, dass sie wirklich mit mehreren Maustasten arbeitet. Dies geschieht in den Systemeinstellungen von OS X und sieht je nach verwendeter Maus etwas anders aus.

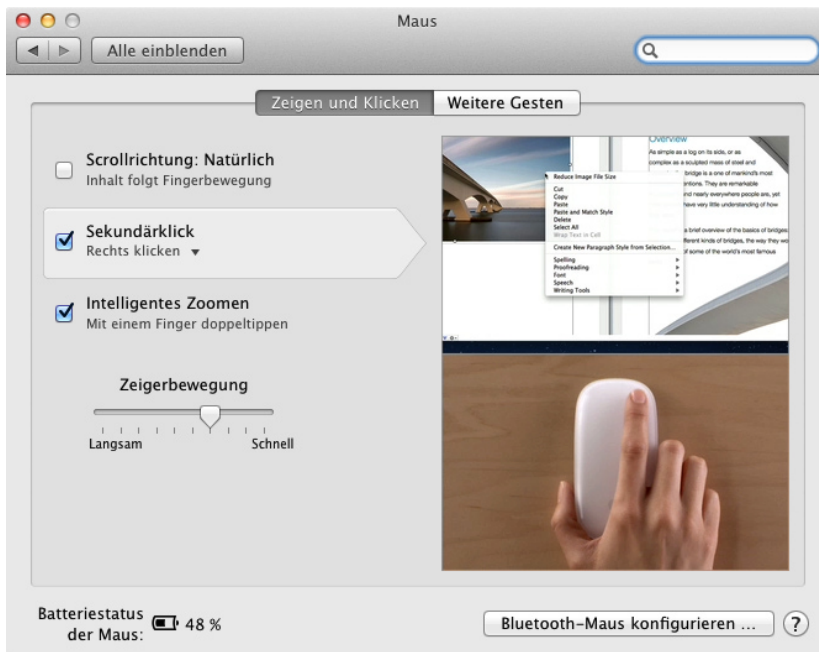
Die Sache mit der Maus



Die Systemeinstellungen für die Apple Mouse:



Die gleichen Einstellungen für die kabellose Magic Mouse:





Wenn dir aber die klassische Arbeitsweise lieber ist, kannst du auch weiterhin mit nur einer Maustaste arbeiten. Für die alternativen Funktionen musst du beim Anklicken immer zusätzlich die `Ctrl`-Taste gedrückt halten. Immer wenn in diesem Buch von der sekundären oder »rechten« Maustaste die Rede ist, kannst du das Gleiche auch bei gedrückter `Ctrl`-Taste mit der primären Maustaste erreichen.

1



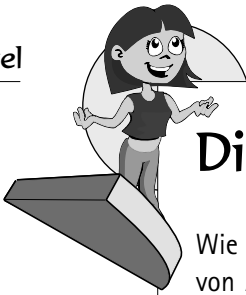
Xcode und dein erstes Projekt

Jetzt geht es los mit dem Programmieren! Zunächst benötigst du aber die passenden Programmierwerkzeuge. Die musst du zwar erst einmal installieren, aber das ist nicht schwer.

In diesem Kapitel lernst du

- ⦿ wie du Xcode bekommst und installierst
- ⦿ wie du eine Projektvorlage auswählst
- ⦿ wie du im Workspace-Fenster den Überblick behältst
- ⦿ wie du Anweisungen schreibst und Hilfe bekommst
- ⦿ wie du Texte auf der Konsole ausgibst
- ⦿ wie du Xcode individuell einrichtest

1



Die Installation von Xcode

Wie im vorherigen Kapitel schon angedeutet, kannst du Xcode kostenlos von Apple direkt aus dem Mac App Store bekommen und benötigst dafür nichts anderes als eine Apple ID, wie du sie vielleicht schon für iTunes oder den iOS App Store verwendest. Es spricht aber auch nichts dagegen, für dich eine neue Apple ID anzulegen, falls du noch keine hast.

Den Mac App Store erreichst du jederzeit über das Apfel-Menü (ganz oben links auf deinem Bildschirm) und Xcode findest du am einfachsten über die Suche des Store Programms. Die Übertragung auf deinen Rechner kann je nach Auslastung des App Store mehr als eine Stunde dauern, da das Paket eine sehr große Datei ist.



Xcode unterscheidet sich ein wenig von den anderen Programmen aus dem Mac App Store, denn nach dem Herunterladen hast du nicht sofort das richtige Programm auf deinem Computer, sondern findest im OS X Launch Pad lediglich das Installationsprogramm **Install Xcode**, das du erst noch durch Anklicken ausführen musst. Dies ist die endgültige Installation von Xcode und auch sie kann ein paar Minuten dauern. Ein wenig Geduld musst du noch haben, es ist aber fast geschafft.

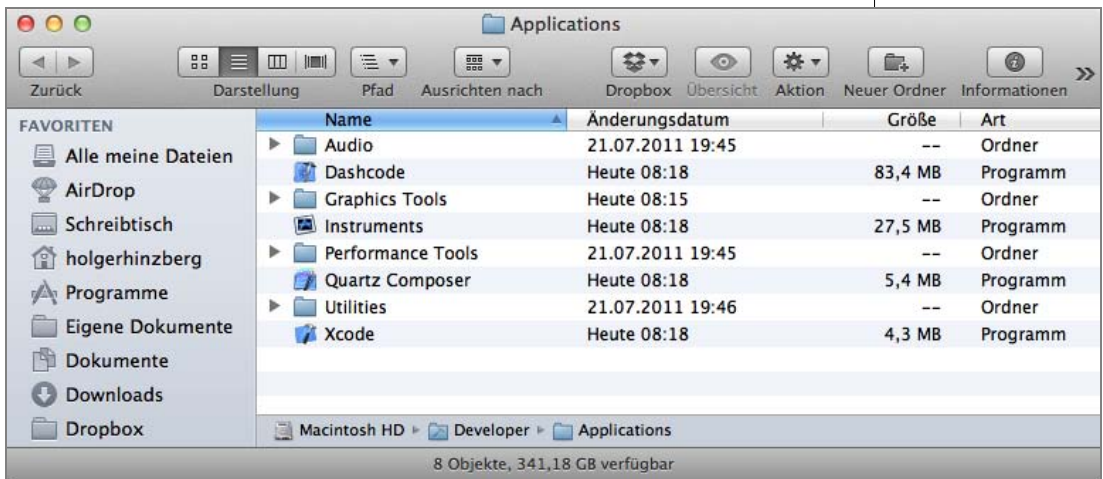


Die Installation von Xcode

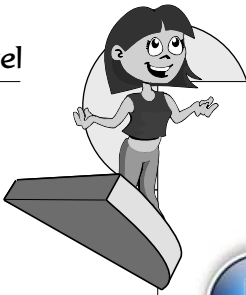


Eventuell musst du dich während der Installation als Administrator anmelden, um fortfahren zu können. In diesem Fall wird sich ein kleines Fenster öffnen, in dem du den Benutzernamen und das Passwort eines Administrators für den Computer eingeben musst. Bei der Installation brauchst du administrative Rechte, später beim Programmieren aber nicht mehr. Ist auch diese Hürde genommen, geht der Rest dann automatisch. Jetzt heißt es abwarten. Nach einigen Minuten ist der Vorgang abgeschlossen.

Auch nach der Installation verhält sich Xcode nicht ganz so wie die anderen Anwendungen auf deinem Mac, denn die Software befindet sich nicht im Ordner Programme, sondern versteckt sich im neu angelegten Ordner Developer/Applications.



Um Xcode in Zukunft leichter und schneller starten zu können, kannst du das Xcode-Symbol aus dem Finder in das Dock ziehen, um so eine Verknüpfung zu erzeugen. Xcode ist der Startpunkt für alle Programme, die du in diesem Buch kennenlernen wirst. Das vom Mac App Store im Dock hinterlegte Symbol des Installationsprogramms brauchen wir hingegen nicht mehr und es kann entfernt werden. Die heruntergeladene Datei Install Xcode im Ordner Programme, die immerhin mehr als drei Gigabyte groß ist, nimmt auch nur unnötigen Festplattenplatz weg und kann ebenfalls gelöscht werden. Möchtest du Xcode nicht im Dock haben, kannst du auch jederzeit die Spotlight-Suche von OS X verwenden, um das Programm zu starten.



Ist Xcode installiert, sind die Vorbereitungen abgeschlossen und es kann mit dem Programmieren losgehen. Schon bald wirst du dein erstes eigenes Programm schreiben.

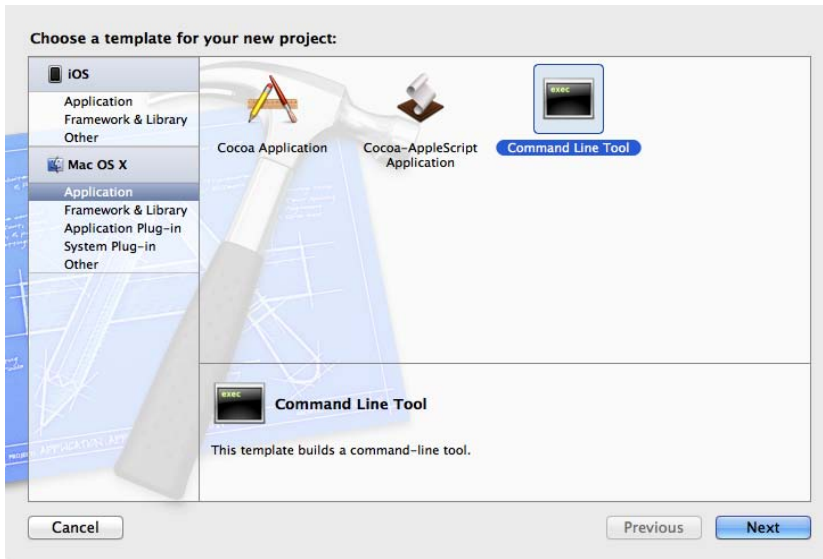
Eine Projektvorlage auswählen

Xcode ist ein sehr mächtiges Werkzeug, mit dem du viele verschiedene Arten von Programmen erzeugen kannst. Die meisten dieser Programme bestehen aus mehreren Einzelteilen, die in sogenannten Projekten verwaltet und zusammengefasst werden.

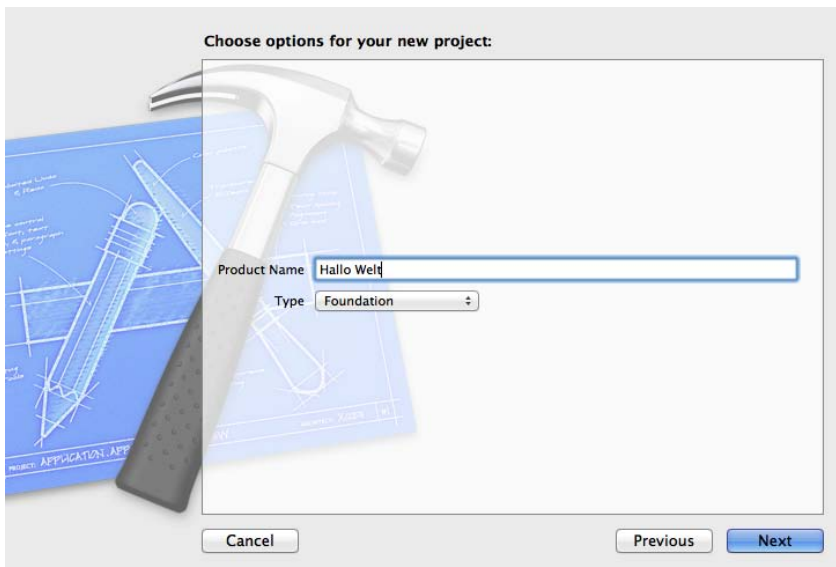
Ein Projekt enthält alle Dateien, aus denen Xcode das fertige Programm erzeugt. Das ist natürlich zum einen der Programmcode, den du selbst geschrieben hast, aber vielleicht auch zusätzlicher Code von anderen Programmierern. Auch weitere Dateien, wie zum Beispiel Bilder, können in einem Projekt enthalten sein. Außerdem gibt es natürlich Informationen darüber, wie Xcode diese einzelnen Teile zusammenbauen soll. Für dein erstes eigenes Programm brauchst du deshalb zuerst ein Projekt.

Wenn du Xcode zum ersten Mal startest, erscheint automatisch das Begrüßungsfenster `WELCOME TO XCODE`, das du aber ignorieren und schließen kannst. Klicke anschließend in der Menüleiste auf `FILE` (Datei) und dann auf `NEW` und `NEW PROJECT` (Neues Projekt). Das Fenster, das sich öffnet, ist die Auswahl der Projektvorlagen und, wie du siehst, gibt es sehr viele verschiedene Projektarten. Dein erstes Programm soll etwas ganz Einfaches sein.

- Klicke in der linken Liste auf den Eintrag `APPLICATION` in der Gruppe `MAC OS X`. Das ist die Oberkategorie für das Projekt.
- Wähle in der rechten Ansicht die Projektvorlage `COMMAND LINE TOOL`.
- Mit `NEXT` oder einem Doppelklick auf das Symbol `COMMAND LINE TOOL` bestätigst du deine Auswahl.

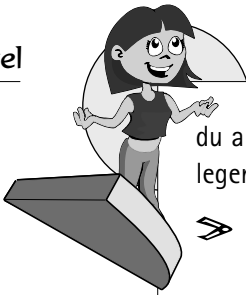


Im nächsten Fenster kannst du weitere Einstellungen vornehmen, so fragt dich Xcode unter anderem, wie das Projekt heißen soll. Den vorgeschlagenen Projekttyp FOUNDATION musst du nicht ändern. Was dieser Typ genau bedeutet, wirst du später noch im Detail erfahren. Trage nur in das Eingabefeld PRODUCT NAME den Namen `Hallo Welt` ein und bestätige das Fenster mit NEXT.



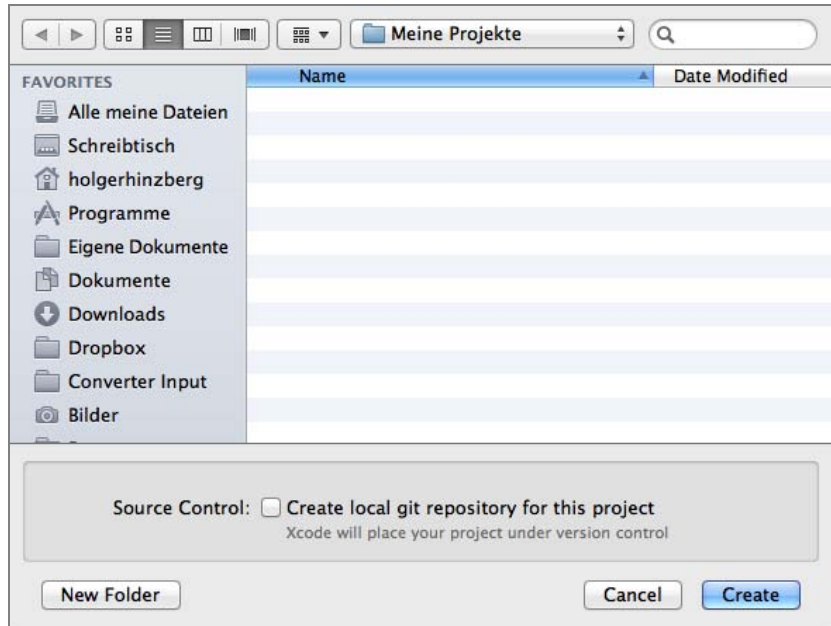
Jetzt möchte Xcode wissen, wo die Dateien gespeichert werden sollen. Immer, wenn du ein Projekt erzeugst, wird ein Ordner erstellt, der genauso heißt wie das Projekt, und in dem alle zugehörigen Dateien abgelegt werden. Das kann sehr schnell unübersichtlich werden, ganz besonders, wenn

1



du an mehreren Projekten arbeitest. Besser ist es, zusätzliche Ordner anzulegen, in denen du Projekte organisieren kannst.

- Klicke auf NEW FOLDER (Neuer Ordner) und erzeuge einen Ordner mit dem Namen `Meine Projekte`.
- Bestätige die Auswahl mit der Schaltfläche CREATE und Xcode wird in `Meine Projekte` einen neuen Ordner mit dem Projektnamen `Hallo Welt` erstellen.



Das Xcode-Menü FILE (Datei) funktioniert wie bei den meisten anderen Programmen. Mit NEW PROJECT erzeugst du ein neues Projekt, das du mit SAVE jederzeit speichern kannst. Mit OPEN kannst du gespeicherte Projekte wieder laden und OPEN RECENT PROJECT zeigt dir eine Liste der zuletzt verwendeten Projekte.

Das Workspace-Fenster

Nun sind die Vorbereitungen abgeschlossen und ein anderes Fenster öffnet sich. Es ist die Xcode-Projektübersicht, auch Workspace-Fenster genannt. Auf den ersten Blick sieht das sehr kompliziert aus, aber lass dich davon nicht abschrecken: Im Laufe dieses Buches wirst du die Funktionen nach und nach kennenlernen.