

Vorwort	xv
1 Grundlagen	1
1.1 Einleitung	1
1.1.1 An wen richtet sich dieses Buch?	1
1.1.2 Welche Vorkenntnisse werden benötigt?	1
1.1.3 Wie arbeitet man am effektivsten mit diesem Buch?	2
1.1.4 Geduld, Motivation und gelegentliche Tiefschläge	3
1.1.5 Das Begleitmaterial zum Buch	4
1.1.6 Fragen zum Buch	4
1.2 Die Programmiersprache C++	4
1.2.1 Von Lochkarten zu C++	5
1.2.2 Objektorientiertes Programmieren	6
1.2.3 Der ANSI-Standard	6
1.2.4 Warum gerade C++?	7
1.3 Jetzt geht es los ... unser erstes Programm	8
1.3.1 Kommentare im Quelltext	9
1.3.2 Die #include-Anweisung	10
1.3.3 Die main-Funktion	12
1.3.4 „cout“ und einige mögliche Escape-Zeichen	13
1.4 Die Entwicklungsumgebung Visual Studio 2012 Express Edition	14
1.4.1 Anlegen eines neuen Arbeitsbereiches	14
1.4.2 Das Programm mithilfe des Quelltexteditors eingeben	16
1.4.3 Laden der Programmbeispiele	17
1.4.4 Das Programm kompilieren und linken	18
1.4.5 Ausführen des Programms	19
1.5 Die Entwicklungsumgebung Xcode	20
1.5.1 Anlegen eines neuen Projekts	21
1.5.2 Hinzufügen und Erstellen von neuen Dateien	23
1.5.3 Das Programm mithilfe des Quelltexteditors eingeben	23
1.5.4 Laden der Programmbeispiele	24

1.5.5 Das Programm kompilieren und linken	24
1.5.6 Ausführen des Programms	26
1.6 Umstellen der Build-Konfiguration	27
1.6.1 Build-Konfiguration in Visual Studio 2012 Express umstellen	27
1.6.2 Build-Konfiguration in Xcode umstellen	27
1.6.3 Der Unterschied zwischen Debug und Release	27
1.7 Aufgabe	28
2 Variablen	31
2.1 Was sind Variablen, und wozu dienen sie?	31
2.2 Datentyp, Variablenname und Wert	31
2.3 Deklarieren und Definieren von Variablen	32
2.4 Rechnen mit Variablen	35
2.4.1 Weitere Rechenoperatoren	37
2.5 Die verschiedenen Datentypen	38
2.6 Namenskonventionen	40
2.7 Konstanten	42
2.7.1 Konstanten mit „const“ erzeugen	42
2.7.2 Konstanten mit „#define“ erzeugen	43
2.7.3 Konstanten mit „enum“ erzeugen	43
2.7.4 Welche der drei Möglichkeiten ist die beste?	44
2.8 Mach mal Platz: Speicherbedarf der Datentypen	45
2.8.1 Überlauf von Variablen	46
2.9 Eingabe von Werten mit „cin“	48
2.10 Casting: Erzwungene Typenumwandlung	49
2.10.1 Casting im C-Stil	51
2.10.2 Casting mit C++	52
2.11 Fehlerquelltext	53
2.11.1 Was soll das Programm eigentlich tun?	54
2.11.2 Lösung zum Fehlerquelltext	55
3 Schleifen und Bedingungen	59
3.1 Was sind Schleifen und Bedingungen, und wozu dienen sie?	59
3.2 Boolesche Operatoren (==, <, >, !=)	60
3.3 Die if-Bedingung	61
3.4 Mittels „eise“ flexibler verzweigen	63
3.5 else if und verschachtelte if-Bedingungen	65
3.6 Logische Operatoren	69
3.7 Verzweigen mit switch und case	71
3.8 Immer und immer wieder: for-Schleifen	74
3.8.1 Initialisierungsteil	76
3.8.2 Bedingungsteil	76

3.8.3 Aktionsteil	77
3.8.4 Zusammenfassung	77
3.9 Eine weitere Rechenoperation: Modulo	77
3.10 Aufgabenstellung	79
3.10.1 Wie geht man an die Aufgabe heran?	79
3.10.2 Lösungsvorschlag	80
3.11 Schleifen mit while und do-while	82
3.12 Verschachtelte Schleifen	85
3.13 Fehlerquelltext	86
3.13.1 Was soll das Programm eigentlich tun?	87
3.13.2 Lösung zum Fehlerquelltext	88
4 Funktionen	91
4.1 Was sind Funktionen, und wozu dienen sie?	91
4.2 Aufbau des Funktionskopfes	93
4.2.1 Rückgabetyp	93
4.2.2 Funktionsname	93
4.2.3 Parameterliste	94
4.3 Aufrufen einer Funktion	94
4.3.1 Funktionsprototypen	94
4.4 Gültigkeitsbereiche	96
4.4.1 Lokale Variablen	97
4.4.2 Globale Variablen	98
4.4.3 Das wäre ja zu einfach gewesen: globale Variablen am Pranger	98
4.5 Verwenden der Funktionsparameter	99
4.5.1 Der Stack	101
4.6 inline-Funktionen	103
4.7 Wann setzt man Funktionen ein?	105
4.7.1 Und wann soll's inline sein?	105
4.8 Überladene Funktionen	106
4.9 Aufgabenstellung	110
4.9.1 Wie geht man an die Aufgabe heran?	111
4.9.2 Lösungsvorschlag	111
4.10 Der sinnvolle Aufbau des Quellcodes	113
4.11 Erstellen und Hinzufügen der neuen Dateien	114
4.12 Das Schlüsselwort „extern“	117
4.13 Ein kleines Spiel: Zahlenraten	118
4.13.1 Zufallszahlen und Bibliotheken	123
4.13.2 Die Hauptfunktion (main)	125
4.13.3 Die Funktion „WaehleLevel“	126
4.13.4 Die Funktion „Spielen“	127
4.13.5 Was gibt es an diesem Listing zu kritisieren?	128

5 Arrays und Strukturen	129
5.1 Was sind Arrays, und wozu dienen sie?	129
5.2 Ein Array erzeugen	129
5.3 Ein Array gleichzeitig deklarieren und definieren	131
5.4 Fehler beim Verwenden von Arrays	133
5.5 char-Arrays	134
5.6 Eingabe von Strings über die Tastatur	136
5.7 Mehrdimensionale Arrays	137
5.8 Arrays und Speicherbedarf	139
5.9 Was sind Strukturen, und wozu dienen sie?	140
5.10 Spielerverwaltung mit Strukturen und Arrays	142
5.11 Aufgabenstellung	145
5.11.1 Wie geht man an die Aufgabe heran?	147
5.11.2 Lösungsvorschlag	147
6 Zeiger und Referenzen	153
6.1 Was sind Zeiger, und wozu dienen sie?	153
6.1.1 Der Stack	155
6.1.2 Vom Flur in den Keller	156
6.2 Die Adresse einer Variablen	157
6.3 Die Adresse einer Variablen in einem Zeiger speichern	158
6.3.1 Schreibweisen bei der Deklaration	160
6.4 Variablen mittels Zeigern ändern	161
6.5 Schön und gut, aber wozu wird das gebraucht?	162
6.6 Noch einmal zurück zum Flur	165
6.7 Was sind Referenzen, und wozu dienen sie?	167
6.7.1 Mit Referenzen arbeiten	168
6.7.2 Regeln bei der Verwendung von Referenzen	170
6.8 Referenzen als Funktionsparameter	171
6.9 Warum Zeiger nehmen, wenn es Referenzen gibt?	172
6.10 Aufgabenstellung	174
6.10.1 Wie geht man an die Aufgabe heran?	175
6.10.2 Lösungsvorschlag	176
7 Klassen	181
7.1 Was sind Klassen, und wozu dienen sie?	181
7.2 Eine einfache Klasse erzeugen und verwenden	183
7.3 Ordnung muss sein	186
7.4 Jetzt wird es privat	189
7.4.1 Private Membervariablen	189
7.4.2 Private Membervariablen und Performance	192
7.4.3 Private Memberfunktionen	192

7.5	Konstruktoren und Destruktoren	193
7.5.1	Der Konstruktor	193
7.5.2	Konstruktoren mit Parameterliste	196
7.5.3	Überladene Konstruktoren	198
7.6	Der Destruktor	200
7.7	Speicherreservierung	202
7.7.1	New und Delete	203
7.7.2	Ein sinnvolleres Beispiel	204
7.7.3	Friss mich, ich bin Dein Speicher	208
7.8	Aufgabenstellung	209
7.8.1	Wie geht man an die Aufgabe heran?	210
7.8.2	Lösungsvorschlag	211
7.9	Vererbung	213
7.9.1	Überschreiben von Memberfunktionen	219
7.9.2	Virtuelle Memberfunktionen	222
7.9.3	Vererbung und Performance	227
7.10	Statische Membervariablen	228
8	Fortgeschrittene Themen	233
8.1	Über dieses Kapitel	233
8.2	printf und sprintf_s	233
8.2.1	printf	234
8.2.2	sprintf_s	236
8.3	Templates	238
8.3.1	Template-Funktionen	238
8.3.2	Template-Klassen	241
8.4	Singletons	245
8.4.1	Eine Klasse für Singletons	245
8.4.2	Einsatz von Singletons	247
8.5	Dateien: Ein- und Ausgabe	250
8.5.1	Werte in eine Datei schreiben und auslesen	250
8.5.2	So viel Zeit muss sein: Fehlerabfrage	253
8.5.3	Instanzen von Klassen in Dateien schreiben	254
8.5.4	Weitere Flags und ihre Bedeutung	256
8.6	Eine nützliche Logfile-Klasse	257
8.6.1	Die Header-Datei der Logfile-Klasse	257
8.6.2	Die Implementierung der Logfile-Klasse	260
8.6.3	Anwendung der Logfile-Klasse	266
8.7	Try, Catch und Assert	268
8.7.1	Das Makro „assert“	269
8.7.2	Fang mich, wenn Du kannst: try und catch	272

8.8	Der Debugger	274
8.8.1	Das Programm im Einzelschrittmodus durchlaufen	274
8.8.2	Haltepunkte und Funktionsaufrufe	278
8.9	SAFE_DELETE – ein nützliches Makro	281
9	Die STL	283
9.1	STL – was ist das?	283
9.1.1	Vektoren	283
9.1.2	Verkettete Listen	290
9.1.3	Strings	299
9.1.4	Maps und Multimaps	306
10	Grundlagen der Windows-Programmierung	315
10.1	Raus aus der Konsole, rein ins Fenster	315
10.1.1	Anlegen eines Win32-Projektes	316
10.1.2	Ein Windows-Grundgerüst	316
10.1.3	Die WinMain-Funktion	320
10.1.4	Die Callback-Funktion	328
10.1.5	Zusammenfassung	329
10.1.6	Ein kurzer Abstecher: Funktionszeiger	330
10.2	Aufgabenstellung	333
10.2.1	Wie geht man an die Aufgabe heran?	334
10.2.2	Lösungsvorschlag	335
10.3	Ein bisschen mehr Interaktion	336
10.3.1	Statischer Text	336
10.3.2	Buttons und Editboxen	338
10.3.3	Messageboxen	339
10.4	Alles noch einmal zusammen	341
11	Sonst noch was?	351
11.1	Um was geht es in diesem Kapitel?	351
11.2	Standardwerte für Funktionsparameter	352
11.3	Memberinitialisierung im Konstruktor	354
11.4	Der this-Zeiger	360
11.5	Der Kopierkonstruktor	365
11.6	Überladen von Operatoren	371
11.7	Mehrfachvererbung	376
11.8	Friend-Klassen	381
12	Ein Spiel mit der SDL	387
12.1	Die SDL – was ist das?	387
12.2	Erstellen und Einrichten des Projekts unter Visual Studio Express 2012 ..	388
12.2.1	Das Projekt anlegen und einrichten	389
12.2.2	Die restlichen Quellcode-Dateien und die .dll-Datei	390

12.3	Erstellen und Einrichten des Projekts unter Xcode	390
12.3.1	Das Projekt anlegen und einrichten	391
12.3.2	Die restlichen Quellcode-Dateien	392
12.4	Projektübersicht	393
12.4.1	Warum plötzlich Englisch? Und wo sind die Zeilennummern?	394
12.4.2	Übersicht der Klassen	394
12.5	Die Implementierung des Spiels	395
12.5.1	Die main-Funktion des Spiels	396
12.5.2	Zeit ist wichtig: Die Klasse CTimer	398
12.5.3	Die Klasse CFramework	400
12.5.4	Bunte Bilder: Die Klasse CSprite	407
12.5.5	Feuer frei: die Klasse CShot	416
12.5.6	Die Klasse CAsteroid	419
12.5.7	Die Hauptfigur: Die Klasse CPlayer	422
12.5.8	Die Klasse CGame	429
12.6	Erweiterungsmöglichkeiten	438
13	Der Einstieg in die Szene	441
13.1	Wie geht's nun weiter?	441
13.2	Die Szene ... was ist das eigentlich?	442
13.3	Welche Möglichkeiten gibt es?	442
13.4	Foren benutzen	443
13.4.1	Ich sag Dir, wer ich bin	443
13.4.2	Richtig posten	444
13.4.3	FAQs und die Suchfunktion	446
13.4.4	Die Kunst zu lesen	447
13.4.5	Selbst Initiative ergreifen und anderen helfen	447
13.5	Weiterbildung mit Tutorials	448
13.6	Anlegen einer Linkssammlung	449
13.7	Copy & Paste	449
13.8	Die Sprache neben C++: Englisch	450
13.9	Auf dem Weg zum eigenen Spiel	451
13.9.1	Mein erstes Spiel: ein 3D-Online-Rollenspiel für 500 Leute	451
13.9.2	Teammitglieder suchen	452
13.9.3	Das fertige Spiel bekannt machen	452
13.9.4	Besuchen von Events zum Erfahrungsaustausch	453
13.10	return 0;	453
Index	455	