

carsten SEIFERT
jan WISLAUG

**Best-
seller**

3. Auflage

SPIELE ENTWICKELN MIT Unity

2D- UND 3D-GAMES MIT **UNITY** UND **C#**
FÜR **DESKTOP, WEB & MOBILE**



Aktualisiert auf Unity 5.6

HANSER

»Lässt von der Programmierung bis zur Umsetzung
von Spielekonzepten keine Fragen offen.« c't

Bleiben Sie auf dem Laufenden!



Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter



www.hanser-fachbuch.de/newsletter



Hanser Update ist der IT-Blog des Hanser Verlags mit Beiträgen und Praxistipps von unseren Autoren rund um die Themen Online Marketing, Webentwicklung, Programmierung, Softwareentwicklung sowie IT- und Projektmanagement. Lesen Sie mit und abonnieren Sie unsere News unter



www.hanser-fachbuch.de/update



Carsten Seifert
Jan Wislaug

Spiele entwickeln mit Unity 5

2D- und 3D-Games
mit Unity und C#
für Desktop, Web & Mobile

3., aktualisierte Auflage

HANSER

Die Autoren:

Carsten Seifert, Süderbrarup, www.hummelwalker.de

Jan Wislaug, Witten, www.jan-wislaug.de

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autoren und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb 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.

The Unity name, logo, brand and other trademarks or images featured or referred to within this book are licensed from and are the sole property of Unity Technologies. Neither this book, its author nor the publisher is affiliated with, endorsed by or sponsored by Unity Technologies or any of its affiliates.

Bibliografische Information der Deutschen Nationalbibliothek:

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

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2017 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Sylvia Hasselbach

Copy editing: Sandra Gottmann, Münster-Nienberge

Erstellung der Dateien für das Beispiel-Game: Alexej Bodemer, Stuhr, www.alexejbodemer.de

Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de

Umschlagrealisation: Stephan Rönigk

Gesamtherstellung: Kösel, Krugzell

Printed in Germany

Print-ISBN: 978-3-446-45197-1

E-Book-ISBN: 978-3-446-45368-5

Inhalt

Vorwort	XIX
1 Einleitung	1
1.1 Multiplattform-Publishing	1
1.2 Das kann Unity (nicht)	2
1.3 Lizenzmodelle	2
1.4 Aufbau und Ziel des Buches	3
1.5 Weiterentwicklung von Unity	4
1.6 Online-Zusatzmaterial	5
2 Grundlagen	7
2.1 Installation	7
2.2 Oberfläche	7
2.2.1 Hauptmenü	9
2.2.2 Scene View	10
2.2.3 Game View	12
2.2.4 Toolbar	14
2.2.5 Hierarchy	16
2.2.6 Inspector	17
2.2.7 Project Browser	21
2.2.8 Console	23
2.3 Das Unity-Projekt	23
2.3.1 Neues Projekt anlegen	24
2.3.2 Bestehendes Projekt öffnen	25
2.3.3 Projektdateien	26
2.3.4 Szene	26
2.3.5 Game Objects	27
2.3.6 Tags	29
2.3.7 Layer	30
2.3.8 Assets	31
2.3.9 Frames	34
2.4 Das erste Übungsprojekt	34

3	C# und Unity	37
3.1	Die Sprache C#	37
3.2	Syntax	38
3.3	Kommentare	39
3.4	Variablen	39
	3.4.1 Namenskonventionen	39
	3.4.2 Datentypen	40
	3.4.3 Schlüsselwort var	41
	3.4.4 Datenfelder/Array	41
3.5	Konstanten	43
	3.5.1 Enumeration	43
3.6	Typkonvertierung	44
3.7	Rechnen	44
3.8	Verzweigungen	45
	3.8.1 if-Anweisungen	46
	3.8.2 switch-Anweisung	48
3.9	Schleifen	49
	3.9.1 for-Schleife	49
	3.9.2 Foreach-Schleife	50
	3.9.3 while-Schleife	50
	3.9.4 do-Schleife	51
3.10	Klassen	51
	3.10.1 Komponenten per Code zuweisen	52
	3.10.2 Instanziierung von Nichtkomponenten	52
	3.10.3 Werttypen und Referenztypen	54
	3.10.4 Überladene Methoden	55
3.11	Der Konstruktor	55
	3.11.1 Konstruktoren in Unity	56
3.12	Lokale und globale Variablen	56
	3.12.1 Namensverwechslung verhindern mit this	56
3.13	Zugriff und Sichtbarkeit	57
3.14	Statische Klassen und Klassenmember	57
3.15	Parametermodifizierer out/ref	58
3.16	Array-Übergabe mit params	59
3.17	Eigenschaften und Eigenschaftsmethoden	60
3.18	Vererbung	61
	3.18.1 Basisklasse und abgeleitete Klassen	62
	3.18.2 Vererbung und die Sichtbarkeit	62
	3.18.3 Geerbte Methode überschreiben	63
	3.18.4 Zugriff auf die Basisklasse	63
	3.18.5 Klassen versiegeln	64
3.19	Polymorphie	64
3.20	Schnittstellen	65
	3.20.1 Schnittstelle definieren	65

3.20.2	Schnittstellen implementieren	65
3.20.3	Zugriff über eine Schnittstelle	66
3.21	Namespaces	67
3.21.1	Eigene Namespaces definieren	68
3.22	Generische Klassen und Methoden	69
3.22.1	List	69
3.22.2	Dictionary	70
4	Skript-Programmierung	73
4.1	MonoDevelop	73
4.1.1	Hilfe in MonoDevelop	74
4.1.2	Syntaxfehler	74
4.2	Nutzbare Programmiersprachen	75
4.2.1	Warum C#?	76
4.3	Unitys Vererbungsstruktur	76
4.3.1	Object	77
4.3.2	GameObject	77
4.3.3	ScriptableObject	77
4.3.4	Component	77
4.3.5	Transform	78
4.3.6	Behaviour	78
4.3.7	MonoBehaviour	78
4.4	Skripte erstellen	78
4.4.1	Skripte umbenennen	79
4.5	Das Skript-Grundgerüst	80
4.6	Unitys Event-Methoden	80
4.6.1	Update	81
4.6.2	FixedUpdate	81
4.6.3	Awake	82
4.6.4	Start	82
4.6.5	OnGUI	82
4.6.6	LateUpdate	83
4.6.7	Aufruf-Reihenfolge	83
4.7	Komponentenprogrammierung	84
4.7.1	Auf GameObjects zugreifen	84
4.7.2	GameObjects aktivieren und deaktivieren	86
4.7.3	GameObjects zerstören	86
4.7.4	GameObjects erstellen	86
4.7.5	Auf Components zugreifen	87
4.7.6	Components hinzufügen	89
4.7.7	Components entfernen	89
4.7.8	Components aktivieren und deaktivieren	89
4.7.9	Attribute	90
4.8	Zufallswerte	91

4.9	Parallel Code ausführen	92
4.9.1	WaitForSeconds	93
4.10	Verzögerte und wiederholende Funktionsaufrufe mit Invoke	94
4.10.1	Invoke	94
4.10.2	InvokeRepeating, IsInvoking und CancelInvoke	94
4.11	Daten speichern und laden	95
4.11.1	PlayerPrefs-Voreinstellungen	95
4.11.2	Daten speichern	96
4.11.3	Daten laden	97
4.11.4	Key überprüfen	97
4.11.5	Löschen	97
4.11.6	Save	97
4.12	Szeneübergreifende Daten	98
4.12.1	Werteübergabe mit PlayerPrefs	98
4.12.2	Zerstörung unterbinden	100
4.13	Debug-Klasse	102
4.14	Kompilierungsreihenfolge	102
4.14.1	Programmsprachen mischen und der sprachübergreifende Zugriff	103
4.15	Ausführungsreihenfolge	103
4.16	Plattformabhängig Code kompilieren	104
4.17	Eigene Assets mit ScriptableObject	105
4.17.1	Neue ScriptableObject-Subklasse erstellen	105
4.17.2	Instanzen eines ScriptableObjects erstellen	106
5	Objekte in der zweiten und dritten Dimension	109
5.1	Das 3D-Koordinatensystem	109
5.2	Vektoren	110
5.2.1	Ort, Winkel und Länge	111
5.2.2	Normalisieren	112
5.3	Das Mesh	113
5.3.1	Normalenvektor	114
5.3.2	MeshFilter und MeshRenderer	115
5.4	Transform	117
5.4.1	Kontextmenü der Transform-Komponente	117
5.4.2	Objekthierarchien	118
5.4.3	Scripting mit Transform	119
5.4.4	Quaternion	119
5.5	Shader und Materials	120
5.5.1	Der Standard-Shader	121
5.5.2	Texturen	138
5.5.3	UV Mapping	141
5.6	3D-Modelle einer Szene zufügen	142
5.6.1	Primitives	142
5.6.2	3D-Modelle importieren	144

5.6.3	In Unity modellieren	145
5.6.4	Prozedurale Mesh-Generierung	146
5.6.5	Level Of Detail	146
5.7	2D in Unity	148
5.7.1	Sprites	149
5.7.2	SpriteRenderer	154
5.7.3	Parallax Scrolling	157
6	Kameras, die Augen des Spielers	161
6.1	Die Kamera	161
6.1.1	Komponenten eines Kamera-Objektes	163
6.1.2	HDR – High Dynamic Range-Rendering	163
6.1.3	Linearer- und Gamma-Farbraum	166
6.2	Kamerasteuerung	169
6.2.1	Statische Kamera	169
6.2.2	Parenting-Kamera	170
6.2.3	Kamera-Skripte	170
6.3	ScreenPointToRay	172
6.4	Mehrere Kameras	173
6.4.1	Kamerawechsel	173
6.4.2	Split-Screen	174
6.4.3	Einfache Minimap	175
6.4.4	Render Texture	177
6.5	Image Effects	179
6.5.1	Beispiel: Haus bei Nacht	179
6.6	Skybox	181
6.6.1	Mehrere Skyboxen gleichzeitig einsetzen	182
6.6.2	Skybox selber erstellen	183
6.7	Occlusion Culling	184
6.7.1	Occluder Static und Occludee Static	186
6.7.2	Occlusion Culling erstellen	186
7	Licht und Schatten	189
7.1	Environment Lighting	189
7.2	Lichtarten	191
7.2.1	Directional Light	192
7.2.2	Point Light	193
7.2.3	Spot Light	194
7.2.4	Area Light	195
7.3	Schatten	196
7.3.1	Einfluss des MeshRenderers auf Schatten	197
7.4	Light Cookies	198
7.4.1	Import Settings eines Light Cookies	198
7.4.2	Light Cookies und Point Lights	199

7.5	Light Halos	200
7.5.1	Unabhängige Halos	201
7.6	Lens Flares	201
7.6.1	Eigene Lens Flares	202
7.7	Projector	202
7.7.1	Standard Projectors	202
7.8	Lightmapping	204
7.8.1	Light Probes	207
7.9	Rendering Paths	209
7.9.1	Forward Rendering	210
7.9.2	Vertex Lit	211
7.9.3	Deferred Lighting	212
7.10	Global Illumination	213
7.10.1	Baked GI	214
7.10.2	Realtime Lighting	215
7.10.3	Lightmapping Settings	216
7.11	Light Explorer	217
7.12	Reflexionen (Spiegelungen)	218
7.12.1	Reflection Probes	219
7.13	Qualitätseinstellungen	222
7.13.1	Quality Settings	222
7.13.2	Qualitätsstufen per Code festlegen	222
8	Physik in Unity	225
8.1	Physikberechnung	225
8.2	Rigidbodyes	226
8.2.1	Rigidbodyes kennenlernen	227
8.2.2	Masseschwerpunkt	228
8.2.3	Kräfte und Drehmomente zufügen	229
8.3	Kollisionen	232
8.3.1	Collider	232
8.3.2	Trigger	236
8.3.3	Static Collider	238
8.3.4	Kollisionen mit schnellen Objekten	238
8.3.5	Terrain Collider	239
8.3.6	Layer-basierende Kollisionserkennung	239
8.3.7	Mit Layer-Masken arbeiten	240
8.4	Wheel Collider	242
8.4.1	Wheel Friction Curve	243
8.4.2	Entwicklung einer Fahrzeugsteuerung	245
8.4.3	Autokonfiguration	252
8.4.4	Fahrzeugstabilität	254
8.5	Physic Materials	254
8.6	Joints	255
8.6.1	Fixed Joint	255

8.6.2	Spring Joint	256
8.6.3	Hinge Joint	256
8.7	Raycasting	256
8.8	Character Controller	258
8.8.1	SimpleMove	258
8.8.2	Move	259
8.8.3	Kräfte zufügen	260
8.8.4	Einfacher First Person Controller	261
8.9	2D-Physik	263
8.9.1	OnCollision2D- und OnTrigger2D-Methoden	265
8.9.2	2D Physic Effectors	266
9	Maus, Tastatur, Touch	269
9.1	Virtuelle Achsen und Tasten	269
9.1.1	Der Input-Manager	269
9.1.2	Virtuelle Achsen	271
9.1.3	Virtuelle Tasten	271
9.1.4	Steuern mit Mauseingaben	272
9.1.5	Joystick-Inputs	272
9.1.6	Anlegen neuer Inputs	273
9.2	Achsen- und Tasteneingaben auswerten	273
9.2.1	GetAxis	273
9.2.2	GetButton	274
9.3	Tastatureingaben auswerten	275
9.3.1	GetKey	275
9.3.2	anyKey	275
9.4	Mauseingaben auswerten	276
9.4.1	GetMouseButton	276
9.4.2	Mauseingaben auf Objekten per Event	277
9.4.3	mousePosition	277
9.4.4	Mauszeiger ändern	278
9.5	Touch-Eingaben auswerten	280
9.5.1	Der Touch-Typ	280
9.5.2	Input.touches	281
9.5.3	TouchCount	281
9.5.4	GetTouch	281
9.5.5	CrossPlatformInput	282
9.6	Beschleunigungssensor auswerten	283
9.6.1	Input.acceleration	284
9.6.2	Tiefpass-Filter	285
9.7	Steuerungen bei Mehrspieler-Games	286
9.7.1	Split-Screen-Steuerung	286
9.7.2	Netzwerkspiele	287

10	Audio	289
10.1	AudioListener	289
10.2	AudioSource	290
	10.2.1 Durch Mauern hören verhindern	292
	10.2.2 Sound starten und stoppen	294
	10.2.3 Temporäre AudioSource	295
10.3	AudioClip	296
	10.3.1 Länge ermitteln	296
10.4	Reverb Zone	296
10.5	Filter	298
10.6	Audio Mixer	298
	10.6.1 Das Audio Mixer-Fenster	298
	10.6.2 Audiosignalwege	302
	10.6.3 Mit Snapshots arbeiten	306
	10.6.4 Views erstellen	307
	10.6.5 Parameter per Skript bearbeiten	307
11	Partikeleffekte mit Shuriken	311
11.1	Editor-Fenster	312
11.2	Particle Effect Control	313
11.3	Numerische Parametervarianten	313
11.4	Farbparameter-Varianten	314
11.5	Default-Modul	314
11.6	Effekt-Module	316
	11.6.1 Emission	316
	11.6.2 Shape	316
	11.6.3 Velocity over Lifetime	318
	11.6.4 Limit Velocity over Lifetime	318
	11.6.5 Inherit Velocity	319
	11.6.6 Force over Lifetime	319
	11.6.7 Color over Lifetime	319
	11.6.8 Color by Speed	320
	11.6.9 Size over Lifetime	320
	11.6.10 Size by Speed	320
	11.6.11 Rotation over Lifetime	320
	11.6.12 Rotation by Speed	321
	11.6.13 External Forces	321
	11.6.14 Noise	321
	11.6.15 Collision	322
	11.6.16 Triggers	323
	11.6.17 Sub Emitter	325
	11.6.18 Texture-Sheet-Animation	325
	11.6.19 Lights	326
	11.6.20 Trails	326
	11.6.21 Renderer	327

11.7	Partikelemission starten, stoppen und unterbrechen	329
11.7.1	Play	330
11.7.2	Stop	330
11.7.3	Pause	330
11.7.4	enableEmission	330
11.8	OnParticleCollision	331
11.8.1	GetCollisionEvents	331
11.9	Feuer erstellen	332
11.9.1	Materials erstellen	332
11.9.2	Feuer-Partikelsystem	333
11.9.3	Rauch-Partikelsystem	336
11.10	Wassertropfen erstellen	340
11.10.1	Tropfen-Material erstellen	340
11.10.2	Wassertropfen-Partikelsystem	341
11.10.3	Kollisionspartikelsystem	343
11.10.4	Kollisionsound	345
12	Landschaften gestalten	347
12.1	Was Terrains können und wo die Grenzen liegen	348
12.2	Terrainhöhe verändern	348
12.2.1	Pinsel	349
12.2.2	Oberflächen anheben und senken	349
12.2.3	Plateaus und Schluchten erstellen	350
12.2.4	Oberflächen weicher machen	351
12.2.5	Heightmaps	351
12.3	Terrain texturieren	353
12.3.1	Textur-Pinsel	354
12.3.2	Texturen verwalten	354
12.4	Bäume und Sträucher	356
12.4.1	Bedienung des Place Tree-Tools	357
12.4.2	Wälder erstellen	357
12.4.3	Mit Bäumen kollidieren	357
12.5	Gräser und Details hinzufügen	358
12.5.1	Detail-Meshs	359
12.5.2	Gräser	360
12.5.3	Quelldaten nachladen	360
12.6	Terrain-Einstellungen	361
12.6.1	Base Terrain	361
12.6.2	Resolution	361
12.6.3	Tree & Details Objects	362
12.6.4	Wind Settings	362
12.6.5	Zur Laufzeit Terrain-Eigenschaften verändern	363
12.7	Der Weg zum perfekten Terrain	364
12.8	Gewässer	365

13	Wind Zones	367
13.1	Spherical vs. Directional	368
13.2	Wind Zone - Eigenschaften	369
13.3	Frische Brise	370
13.4	Turbine	370
14	GUI	371
14.1	Das UI-System uGUI	372
14.1.1	Canvas	372
14.1.2	RectTransform	376
14.1.3	UI-Sprite Import	380
14.1.4	Grafische Controls	381
14.1.5	Interaktive Controls	385
14.1.6	Controls designen	392
14.1.7	Animationen in uGUI	393
14.1.8	Event Trigger	394
14.2	Screen-Klasse	395
14.2.1	Schriftgröße dem Bildschirm anpassen	395
14.3	OnGUI-Programmierung	396
14.3.1	GUI	397
14.3.2	GUILayout	399
14.3.3	GUIStyle und GUISkin	400
15	Prefabs	403
15.1	Prefabs erstellen und nutzen	403
15.2	Prefab-Instanzen erzeugen	403
15.2.1	Instanzen per Code erstellen	404
15.2.2	Instanzen weiter bearbeiten	405
15.3	Prefabs ersetzen und zurücksetzen	405
15.4	Prefab-Verbindungen auflösen	406
16	Internet und Datenbanken	407
16.1	Die WWW-Klasse	407
16.1.1	Rückgabewert-Formate	408
16.1.2	Parameter übergeben	409
16.2	Datenbank-Kommunikation	410
16.2.1	Daten in einer Datenbank speichern	410
16.2.2	Daten von einer Datenbank abfragen	411
16.2.3	Rückgabewerte parsen	413
16.2.4	Datenhaltung in eigenen Datentypen	414
16.2.5	HighscoreCommunication.cs	416
16.2.6	Datenbankverbindung in PHP	417

17	Animationen	419
17.1	Allgemeiner Animation-Workflow	420
17.2	Animationen erstellen	420
17.2.1	Animation View	421
17.2.2	Curves vs. Dope Sheet	422
17.2.3	Animationsaufnahme	422
17.2.4	Beispiel Fallgatter-Animation	427
17.3	Animationen importieren	428
17.3.1	Rig	429
17.3.2	Animationen	431
17.4	Animationen einbinden	434
17.4.1	Animator Controller	435
17.4.2	Animator-Komponente	450
17.4.3	Beispiel Fallgatter: Animator Controller	451
17.5	Controller-Skripte	453
17.5.1	Parameter des Animator Controllers setzen	454
17.5.2	Animation States abfragen	454
17.5.3	Beispiel Fallgatter Controller-Skript	455
17.6	Animation Events	457
17.7	Das „alte“ Animationssystem	458
18	Künstliche Intelligenz	461
18.1	NavMeshAgent	462
18.1.1	Eigenschaften der Navigationskomponente	463
18.1.2	Zielpunkt zuweisen	464
18.1.3	Pfadsuche unterbrechen und fortsetzen	464
18.2	Navigation-Fenster	465
18.2.1	Agents Tab	466
18.2.2	Object Tab	467
18.2.3	Bake Tab	467
18.2.4	Areas Tab	468
18.3	NavMeshObstacle	469
18.4	Off-Mesh Link	470
18.4.1	Automatische Off-Mesh Links	470
18.4.2	Manuelle Off-Mesh Links	471
18.5	Point & Click-Steuerung für Maus und Touch	472
19	Fehlersuche und Performance	475
19.1	Fehlersuche	475
19.1.1	Breakpoints	476
19.1.2	Variablen beobachten	477
19.1.3	Console Tab nutzen	478
19.1.4	GUI- und GUILayout nutzen	478
19.1.5	Fehlersuche bei mobilen Plattformen	479

19.2	Performance	481
19.2.1	Rendering-Statistik	482
19.2.2	Batching-Verfahren	483
19.2.3	Analyse mit dem Profiler	484
19.2.4	Echtzeit-Analyse auf Endgeräten	486
20	Spiele erstellen und publizieren	489
20.1	Der Build-Prozess	489
20.1.1	Szenen des Spiels	490
20.1.2	Plattformen	491
20.1.3	Notwendige SDKs	491
20.1.4	Plattformspezifische Optionen	492
20.1.5	Developer Builds	492
20.2	Publizieren	493
20.2.1	App	494
20.2.2	Browser-Game	494
20.2.3	Desktop-Anwendung	495
21	Erstes Beispiel-Game: 2D-Touch-Game	497
21.1	Projekt und Szene	497
21.1.1	Die Kamera	499
21.1.2	Texturen importieren und Sprites definieren	500
21.2	Gespenster und Hintergrund	502
21.2.1	Gespenster animieren	505
21.2.2	Gespenster laufen lassen	509
21.2.3	Gespenster-Prefab erstellen	511
21.3	Der GameController	512
21.3.1	Der Spawner	512
21.3.2	Level-Anzeige	514
21.3.3	Der Input-Controller	515
21.3.4	Game Over-UI	517
21.3.5	Hintergrundmusik	524
21.4	Punkte zählen	525
21.5	Spielende	526
21.6	Spiel erstellen	527
22	Zweites Beispiel-Game: 3D Dungeon Crawler	529
22.1	Level-Design	530
22.1.1	Modellimport	531
22.1.2	Materials konfigurieren	532
22.1.3	Prefabs erstellen	533
22.1.4	Dungeon erstellen	535
22.1.5	Dekoration erstellen	540
22.2	Inventarsystem erstellen	542
22.2.1	Verwaltungslogik	542

22.2.2	Oberfläche des Inventarsystems	550
22.2.3	Inventar-Items	553
22.3	Game Controller	560
22.4	Spieler erstellen	560
22.4.1	Lebensverwaltung	562
22.4.2	Spielersteuerung	573
22.4.3	Wurfstein entwickeln	581
22.4.4	Lautstärke steuern	587
22.5	Quest erstellen	588
22.5.1	Erfahrungspunkte verwalten	588
22.5.2	Questgeber erstellen	590
22.5.3	Sub-Quest erstellen	599
22.6	Gegner erstellen	604
22.6.1	Model-, Rig- und Animationsimport	604
22.6.2	Komponenten und Prefab konfigurieren	605
22.6.3	Animator Controller erstellen	607
22.6.4	NavMesh erstellen	609
22.6.5	Umgebung und Feinde erkennen	610
22.6.6	Gesundheitszustand verwalten	613
22.6.7	Künstliche Intelligenz entwickeln	617
22.7	Eröffnungsszene	626
22.7.1	Szene erstellen	626
22.7.2	Startmenü-Logik erstellen	627
22.7.3	Menü-GUI erstellen	629
22.8	WebGL-Anpassungen	631
22.8.1	WebGL-Input ändern	631
22.8.2	Quit-Methode in WebGL abfangen	632
22.9	Finale Einstellungen	633
22.10	So könnte es weitergehen	636
23	Der Produktionsprozess in der Spieleentwicklung	637
23.1	Die Produktionsphasen	637
23.1.1	Ideen- und Konzeptionsphase	638
23.1.2	Planungsphase	638
23.1.3	Entwicklungsphase	638
23.1.4	Testphase	639
23.1.5	Veröffentlichung und Postproduktion	639
23.2	Das Game-Design-Dokument	639
24	Schlusswort	641
	Index	643

Vorwort

Für viele von uns sind Computerspiele und Handysgames heutzutage allgegenwärtige Wegbegleiter. Egal ob auf dem Smartphone, dem Tablet, installiert auf dem heimischen PC oder direkt aufgerufen im Browser werden sie von uns täglich genutzt. Manchmal dienen sie als Zeitvertreib, bis der nächste Bus kommt, manchmal sind sie aber auch Bestandteil eines intensiven Hobbys.

Aber nicht nur das Spielen kann Spaß machen, auch das Entwickeln dieser Games kann begeistern. Sowohl im Freizeitbereich als auch in der Arbeitswelt wird der Beruf des Spieleentwicklers immer beliebter. Es ist also kein Wunder, dass mittlerweile viele, teilweise sogar staatlich anerkannte, Studiengänge existieren, die sich dem Entwickeln von Computerspielen widmen.

In diesem Buch möchten wir Ihnen Unity, eine weit verbreitete Entwicklungsumgebung für Computerspiele und auch andere Anwendungen, näherbringen. Wir erklären und erläutern ausführlich, wie Sie mit diesem Werkzeug Spiele entwickeln können. Dabei richtet sich das Buch sowohl an Einsteiger und Umsteiger als auch an Spieleentwickler, die mit Unity nun richtig durchstarten möchten.

Als ursprünglicher Autor dieses Buches möchte ich, Carsten Seifert, mich an dieser Stelle ganz besonders bei meiner Frau Cornelia bedanken, die mich während des Schreibens so geduldig unterstützt hat und mir jederzeit beim Formulieren und Korrigieren hilfsbereit zur Seite stand.

Weiter möchte ich Sieglinde Schär, Kristin Rothe und dem gesamten Hanser-Verlag-Team danken, die mir nicht nur das Schreiben dieses Buches ermöglicht haben, sondern auch bei der Arbeit an den ersten beiden Auflagen des Buches jederzeit mit Rat und Tat zur Seite standen.

Auch danke ich ganz herzlich Alexej Bodemer, der für das Beispiel-Game dieses Buches alle 3D-Modelle, Texturen und Musikdateien entworfen und zur Verfügung gestellt hat.

Ich, Jan Wislaug, möchte mich als überarbeitender Autor vor allem bei Sylvia Hasselbach vom Hanser Verlag bedanken. Sie stand mir bei der Arbeit an der dritten Auflage immer freundlich mit Rat und Tat beiseite.

Darüber hinaus danke ich Carsten Seifert für die gute Zusammenarbeit und dafür, dass sich sein Werk dank der guten Struktur so toll überarbeiten ließ.

Zusammen möchten wir beide Will Goldstone und Unity Technologies danken, die uns die bis dato aktuellsten Beta-Versionen zur Verfügung gestellt haben.

Nicht zuletzt danken wir auch der gesamten Community, die zum einen Carsten Seifert auf seinem Blog und seinen sozialen Kanälen begleitet hat und zum anderen Jan Wislaug bei seiner Überarbeitung auf die neue Unity-Version unterstützte.

Carsten Seifert und Jan Wislaug

im Juli 2017

1

Einleitung

Computerspiele gehören heutzutage zu den beliebtesten Freizeitgestaltungen unserer Zeit. Mit Zunahme der Popularität ist aber auch der Anspruch an diese Spiele gestiegen. Während in den ersten Jahren bis Jahrzehnten dieser jungen Branche noch ein einziger Programmierer ausreichte, um alle notwendigen Aufgaben zu erledigen, werden anspruchsvolle Computerspiele heutzutage meist von großen Teams umgesetzt. Hier arbeiten 3D-Modellierer, Grafiker, Sounddesigner, Level-Designern und natürlich auch Programmierer aus unterschiedlichen Sparten Hand in Hand.

Um den stetig wachsenden Ansprüchen zu genügen, sind aber auch die Werkzeuge der Entwickler ständig mitgewachsen. Eines dieser Werkzeuge ist Unity. Unity ist eine Spieleentwicklungsumgebung für Windows- und Mac OS X-Systeme und wird von der aus Dänemark stammenden Firma Unity Technologies entwickelt. Mit ihr können Sie sowohl interaktive 3D- als auch 2D-Inhalte erstellen. Wir sprechen deshalb von Inhalten und nicht nur von Spielen, weil Unity zwar eigentlich für die Entwicklung von 3D-Spielen gedacht war, mittlerweile aber auch immer häufiger Anwendung in anderen Bereichen findet. So wird es beispielsweise für Architekturvisualisierungen genutzt, im E-Learning-Bereich eingesetzt oder in der Digital-Signage-Branche für das Erstellen digitaler Werbe- und Informationssysteme genommen.

Da Unity ursprünglich für die Entwicklung von 3D-Spielen konzipiert wurde, lautet die Internet-Adresse der Firma *unity3d.com*. Dies ist der Grund, weshalb die Entwicklungssoftware auch gerne mal „Unity3D“ genannt wird, was aber eben nicht ganz korrekt ist.

■ 1.1 Multiplattform-Publishing

Eine besondere Stärke von Unity ist die Unterstützung von Multiplattform-Publishing. Das bedeutet, dass Sie in Unity ein Spiel einmal entwickeln können, das Sie dann aber für mehrere Plattformen exportieren können. Aktuell werden mehr als 25 Plattformen unterstützt, die sich grob in folgende Kategorien einteilen lassen:

- Die sogenannten „Standalones“. Hierzu gehören: Windows Desktop, Mac, Linux (und Steam OS)

- Die mobilen Smartphone-/Tablet-Plattformen: iOS, Android, Windows Phone, Fire OS
- Die Spielekonsolen: PlayStation 4, PlayStation Vita, Xbox One, Nintendo Switch, Wii U, Nintendo 3DS
- Die Kategorie „Smart-TV“. Hier werden folgende Systeme unterstützt: AndroidTV, Samsung SMART-TV, tvOS

In Verbindung mit diesen Plattformen bietet Unity die Möglichkeit, Inhalte direkt für verschiedene Virtual-Reality-Systeme zu entwickeln. Hierzu gehören: Oculus-Rift, Google-Cardboard, Steam-VR, Playstation-VR, Gear-VR, Microsoft Hololens und Daydream.

Auch für Web-Inhalte bietet Unity entsprechende Features. Mit der Plattform „WebGL“, die im Browser läuft und auf HTML5 basiert, können Sie Nutzern direkten Zugriff auf Ihre Inhalte geben.

Bei den erwähnten Spielekonsolen müssen Sie allerdings noch eine Einschränkung beachten: Sie müssen hier extra Lizenzen erwerben, die zum einen nicht günstig und zum anderen nur für Firmen verfügbar sind, die von den jeweiligen Konsolenherstellern auch als Entwickler akzeptiert wurden. Deshalb werden wir die Konsolenentwicklung in diesem Buch auch außen vor lassen und nicht näher beleuchten.

■ 1.2 Das kann Unity (nicht)

Unity bringt eine ganze Reihe an nützlichen Werkzeugen mit, um Spiele und andere 2D- und 3D-Anwendungen zu entwickeln. So gibt es neben einer ausgeklügelten Physik-Engine auch Tools für Partikeleffekte, zur Landschaftsgestaltung oder auch für Animationen. Außerdem wird Unity mit einer extra angepassten Version der Softwareentwicklungsumgebung MonoDevelop ausgeliefert, in der die Programmierung umgesetzt und das Debugging vorgenommen werden kann.

Eines ist Unity allerdings nicht: Es ist keine 3D-Modellierungssoftware. Unity bietet zwar von Haus aus einige 3D-Grundobjekte an, sogenannte Primitives, die für kleinere Aufgaben genutzt werden können, aber für richtige Modellierungsaufgaben sollten Sie auf die dafür entwickelten Spezialtools wie 3ds Max oder das kostenlose Blender zurückgreifen.

■ 1.3 Lizenzmodelle

Sie haben die Möglichkeit, auf verschiedene Modelle zurückzugreifen, wenn Sie Unity nutzen möchten. Zum eine gibt es eine komplett kostenfreie Variante, welche bereits alle oben genannten Zielplattformen unterstützt und eine komplette Spieleentwicklung bis hin zum fertigen Spiel erlaubt. Es gibt jedoch auch einige Einschränkungen, die sich allerdings nur auf erweiterte oder visuelle Features beziehen. So müssen Sie zum Beispiel auf die dunkle Oberfläche (Skin) Ihres Unity-Programms verzichten oder haben auch keinen Zugriff auf den Premium-Support von Unity.

Bei der kostenpflichtigen Variante unterscheidet man die drei Abo-Modelle „Plus“, „Pro“ und „Enterprise“, welche jeweils monatlich zu bezahlen sind. Je teurer das Abo ist, umso mehr Funktionen stehen zur Verfügung.

Da sowohl die kostenfreie als auch die kostenpflichtige Edition kommerziell genutzt werden darf, gibt es beim Einsatz der kostenlosen Version die Vorgabe, dass nur Firmen bzw. Entwickler diese einsetzen dürfen, die nicht mehr als 100 000 US-Dollar Umsatz in einem Geschäftsjahr machen.

■ 1.4 Aufbau und Ziel des Buches

Mit diesem Buch werden Sie lernen, auf Basis von Unity eigene 2D- und 3D-Spiele zu entwickeln. Sie werden sich mit den unterschiedlichen Tools der Game Engine vertraut machen und die Skript-Programmierung erlernen. Dabei steht aber nicht das Ziel im Fokus, wirklich jede einzelne Funktion und Möglichkeit zu beleuchten, die Unity dem Entwickler anbietet. Vielmehr zeigen wir Ihnen, wie die unterschiedlichen Bereiche von Unity funktionieren und miteinander zusammenarbeiten. Denn der Funktionsumfang dieser Spieleentwicklungsumgebung ist mittlerweile so umfangreich geworden, dass es gar nicht mehr möglich ist, alle Tools und deren Möglichkeiten bis ins letzte Detail in einem einzigen Buch ausführlich zu behandeln. Daher liegt der Schwerpunkt auf den Kernfunktionen und wichtigen Optimierungstechniken, die in Unity für die 2D- und 3D-Spieleentwicklung bereitgestellt werden, ergänzt um Anwendungsbeispiele und Praxistipps.

Möchten Sie weiter in die Tiefe eines speziellen Tools gehen oder mehr Informationen zu bestimmten Scripting-Klassen erhalten, empfehlen wir Ihnen die mit Unity mitgelieferten Hilfe-Dokumente, die Sie über das Help-Menü erreichen. Dort finden Sie sowohl ein ausführliches Manual über die in Unity integrierten Tools sowie eine *Scripting-Referenz* über alle Unity-Klassen und deren Möglichkeiten. Letztere finden Sie auch über die Hilfe von MonoDevelop, der mitgelieferten Programmierumgebung.

Spieleentwicklung wird häufig auch als Spieleprogrammierung bezeichnet, auch wenn viele Aufgaben in der Spieleentwicklung heutzutage nicht mehr programmiert, sondern mithilfe von Tools erledigt werden. Nichtsdestotrotz ist der Programmieranteil bei der Entwicklung eines Spiels doch immer noch sehr hoch. Deshalb wird auch das Buch zunächst mit zwei größeren programmierbezogenen Kapiteln beginnen. Das erste behandelt allgemeine Grundlagen der Programmierung, das zweite geht auf die Unity-spezifischen Themen ein. Erst danach werden wir in die 3D-Welt von Unity eintauchen und die verschiedenen Werkzeuge behandeln. Der Aufbau ist deshalb so gewählt, weil wir in den folgenden Kapiteln immer wieder kleinere Skript-Beispiele zeigen, die Einsatzmöglichkeiten in der Praxis demonstrieren.

Ganz grundsätzlich ist die Reihenfolge der einzelnen Kapitel so gewählt, dass die Inhalte aufeinander aufbauen. Aus diesem Grund kommen auch erst zum Ende des Buches die beiden Beispiel-Games, eines für 2D und eines für 3D. In diesen werden alle angesprochenen Themen noch einmal aufgegriffen und die gesamten Zusammenhänge in der Praxis gezeigt. Nichtsdestotrotz können Sie gerne diese Kapitel auch vorziehen. Speziell das 2D-Spiel eignet sich hierfür gut.

Im Buch werden wir Hinweiskästen, wie den obigen, einsetzen, um Hinweise und Tipps zu geben. Je nach Typ des Hinweises werden diese mit unterschiedlichen Icons ausgezeichnet.



Praxistipps



Hinweise zu weiterführenden Inhalten im Internet



Allgemeine Hinweise

■ 1.5 Weiterentwicklung von Unity

Die Spieleindustrie gehört zu den Branchen, die sich aktuell am schnellsten verändern. Kein Wunder also, dass auch Unity ständig weiterentwickelt wird und neue Funktionen erhält. Sollten Sie Unterschiede zwischen dem Buch und Ihrer Unity-Version erkennen, wird dies sicher der ständigen Weiterentwicklung von Unity geschuldet sein.

Aber nicht nur der Funktionsumfang wird ständig weiterentwickelt, auch das Lizenzmodell von Unity ist nicht von Veränderungen ausgeschlossen. In den letzten Jahren hat sich dieses zunehmend geändert. Seit es die kostenlose Version von Unity gibt, wurde deren Funktionsumfang gravierend erweitert (genauso wie der der Pro-Version). So standen anfangs nur Basisfunktionalitäten zur Verfügung. Später kamen dann die Mobile-Export-Möglichkeiten hinzu, bis 2015 schließlich auch alle anderen Pro-Features der Engine in die kostenlose Version integriert wurden, wobei noch eine Ausnahme besteht. So lässt die Personal Edition mittlerweile zwar den *Customizable Splash Screen* zu, aber es erscheint vor jedem Spiel, das mit dieser Version erstellt wurde, immer noch der Unity-Schriftzug. Besitzen Sie allerdings Unity Professional, können Sie auch das anpassen.

Aktuell arbeitet Unity mit Facebook zusammen, um die Plattform „Facebook Gameroom“ weiter voranzubringen. Auch der Bereich „Virtual Reality“ wird kontinuierlich ausgebaut. In kommenden Versionen der Software wird sich also immer wieder etwas tun, um ein noch breiteres Spektrum an Funktionen zu bieten. Auf der Firmen-Website von Unity Technologies können Sie sich immer über den aktuellen Entwicklungsstand informieren.

■ 1.6 Online-Zusatzmaterial

Auf einer Unterseite meiner Homepage stelle ich Ihnen einen Bereich zur Verfügung, in dem Sie Zusatzmaterialien erhalten. Im oberen, für jedermann sichtbaren Abschnitt, befindet sich eine Sammlung an kleinen Ausbesserungen und Hinweisen zum Buch. Darunter finden Sie die Beispielprojekte, welche aber nur nach einem erfolgreichen Login sichtbar sind.



Passwortgeschützter Bereich für Zusatzmaterialien zum Buch

URL: http://jan-wislaug.de/Unitybuch_ZusatzMaterialien.htm

Benutzername: Leser

Passwort: **Unity5Buch!**

In dem geschützten Bereich finden Sie Links zu Projektdaten unter anderem für:

- Ein touch-basiertes 2D-Beispiel-Game mit allen dazugehörigen Ressourcen
- Ein 3D-Beispiel-Game (Dungeon Crawler) mit allen dazugehörigen Ressourcen
- Ein Anwendungsbeispiel für eine Auto-Steuerung inklusive eines 3D-Modells mit zusätzlichen Skripten
- Ein Beispiel für eine Sprite-Animation
- Ein Anwendungsbeispiel für Parallax Scrolling
- Eine Vorlage für ein einfaches Übungsprojekt
- Mehrere ergänzende Video-Tutorials

2

Grundlagen

In diesem Kapitel werden Sie die Oberfläche von Unity sowie deren grundlegende Bedienung kennenlernen. Wir werden auf die Struktur eines Unity-Projektes eingehen und die grundlegenden Prinzipien von Unity behandeln.

■ 2.1 Installation

Sollten Sie noch keine Installationsdatei haben, besuchen Sie zunächst die Seite <https://store.unity.com> bzw. <http://www.unity3d.com> und laden sich dort die aktuelle Unity-Version herunter. Nach dem Download können Sie Unity installieren. Da die Installation eigentlich selbsterklärend ist, sollten Sie lediglich darauf achten, dass Sie eine Komplettinstallation machen und keine Teile davon ausnehmen. Profis können natürlich selber bewerten, was sie benötigen, Anfängern würden wir aber immer eine Komplettinstallation empfehlen.

Wenn Sie die Installation abgeschlossen haben und Unity das erste Mal starten, wird sich das Projekt-Fenster von Unity zeigen. Über dieses können Sie ein existierendes Projekt öffnen oder ein neues Projekt erstellen. Außerdem finden Sie in der unteren Leiste des Fensters Web-Links zum Community-Bereich, zur Online-Dokumentation sowie zum Tutorial-Bereich von Unity Technologies.

Mehr zum Öffnen und Erstellen eines Projektes erfahren Sie im Abschnitt „Das Unity-Projekt“. Für den ersten Start können Sie einfach die Standardeinstellungen übernehmen und über **NEW PROJECT** ein neues Projekt erstellen.

■ 2.2 Oberfläche

Die Oberfläche von Unity besteht aus mehreren frei anpassbaren Fenstern (auch Tabs genannt), die sich innerhalb von Unity übereinander und nebeneinander andocken sowie auch außerhalb des Hauptfensters verschieben lassen.

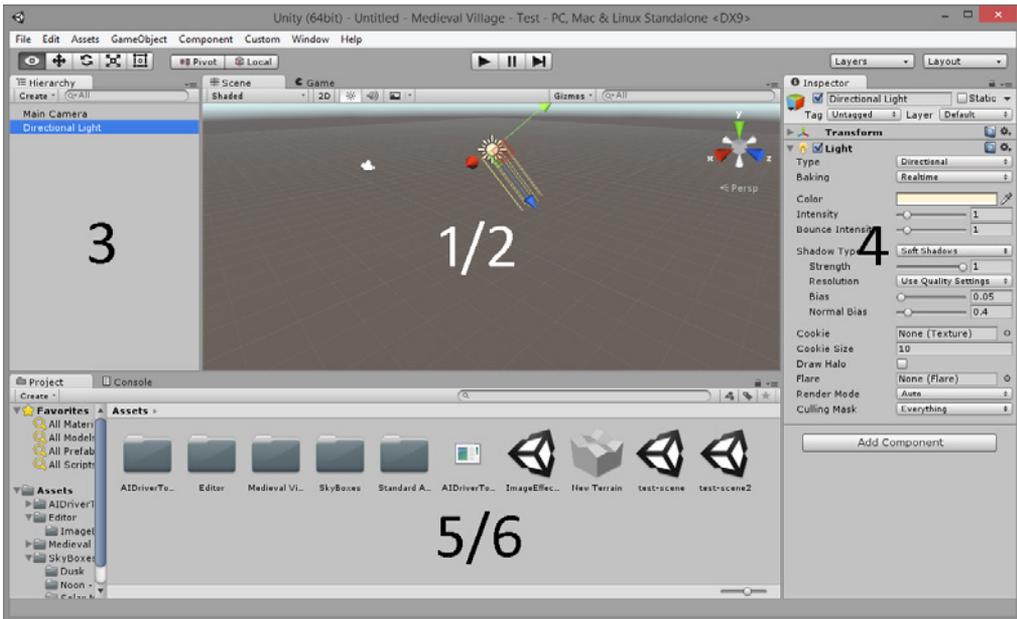


Bild 2.1 Default-Ansicht von Unity

Die Fenster (Tabs) besitzen zwar unterschiedliche Aufgaben, gehören funktional aber alle zusammen und werden deshalb auch kontinuierlich untereinander synchronisiert.

1. **Scene View** dient dem interaktiven Gestalten von Szenen und 3D-Welten.
2. **Game View** dient als Vorschau des fertigen Spiels. Dieses Fenster wird in Bild 2.1 von der *Scene View* verdeckt, wird aber von Unity automatisch nach vorne gebracht, wenn das Spiel gestartet wird.
3. **Hierarchy** zeigt alle in der Szene existierenden Objekte (*GameObjects*) in deren Hierarchiestruktur an.
4. **Inspector** zeigt alle Komponenten und öffentlichen Parameter des aktuell selektierten *GameObjects* an.
5. **Project Browser** dient dem Anzeigen und Verwalten aller digitalen Inhalte (auch *Assets* genannt), die zu dem Projekt gehören.
6. **Console** dient dem Anzeigen von Fehler- und Hinweismeldungen. Dieses Fenster befindet sich in der Default-Ansicht hinter dem *Project Browser*. Um die Meldungen zu sehen, müssen Sie auf den Reiter des Fensters klicken.

Abgesehen von der *Game View* und dem *Console-Tab* können Sie den Tabs nicht nur Informationen entnehmen, sondern auch die Objekte verändern.

Sie können die Tab-Anordnungen in Unity jederzeit speichern und auch zurücksetzen. Hierfür stellt Unity im oberen Hauptmenü über **WINDOW/LAYOUTS** entsprechende Funktionen und Standardanordnungen zur Verfügung. Als Schnellzugriff dient hier das *Layouts-Drop-down-Menü*, das Sie ganz rechts im oberen Bereich von Unity finden.

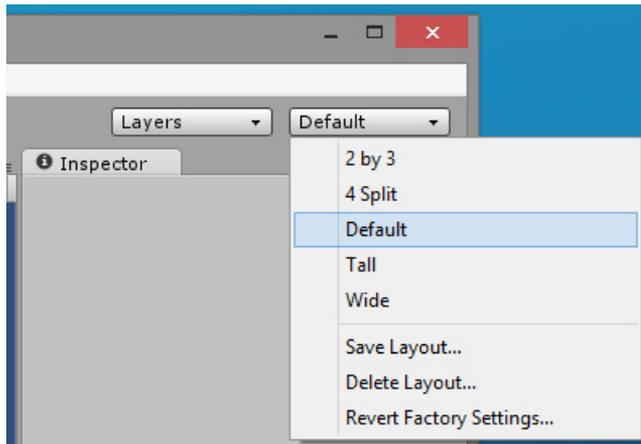


Bild 2.2 Schnellzugriff auf die Layout-Funktionen

Wenn Sie mit **SAVE LAYOUT** eigene Fenster-Anordnungen speichern, stehen diese Layouts nicht nur in diesem Projekt zur Verfügung, sondern auch in allen anderen Unity-Projekten. Mit **DELETE LAYOUT** löschen Sie diese natürlich ebenfalls in allen Projekten. Sollten Sie dabei eine Standardsicht aus Versehen gelöscht haben, können Sie über **REVERT FACTORY SETTINGS** alle Layout-Einstellungen wieder auf die Werkseinstellungen zurücksetzen.

2.2.1 Hauptmenü

Oberhalb aller Subfenster und Toolbars befindet sich das Hauptmenü von Unity. Viele Teile dieses Menüs finden Sie zusätzlich noch einmal als Schnellzugriff-Menüs in anderen Bereichen von Unity wieder, wie z. B. das *Layouts-Drop-down-Menü* (siehe oben).



Bild 2.3 Hauptmenü

Hinter diesen Hauptmenüpunkten verbergen sich folgende Funktionen:

- **File** beinhaltet alle Funktionen, die sich mit dem Erstellen und Speichern von Dateien auf Projekt- und Szene-Ebene beschäftigen.
- **Edit** besitzt vor allem Einstellungen zum Verändern von Daten.
- **Asset** beschäftigt sich mit dem Verwalten und Erstellen neuer *Assets*, z. B. von Skripten und Materialien.
- **GameObject** beschäftigt sich vor allem mit dem Erstellen verschiedener, vorkonfigurierter *GameObjects*.
- **Component** bietet alle Standardkomponenten von Unity an.
- **Window** bietet vor allem weitere Fenster/Tabs an, die Sie anzeigen können.
- **Help** besitzt hauptsächlich Quellen und Links zu weiterführenden Informationen für Unity. Hier gelangen Sie auch zum „Unity Manual“ und zu der „Scripting Reference“, wo Sie viele weiterführende Informationen finden.