

Thomas Theis



Einstieg in Unity

2D- und 3D-Spiele entwickeln

Ideal für
Programmier-
anfänger

- ▶ Schritt für Schritt zum eigenen Spiel
- ▶ Mit C#-Programmierkurs für Unity-Skripte
- ▶ 17 vollständig entwickelte Spiele aus vielen Genres



Alle Beispielprojekte zum Download



Rheinwerk
Computing

Einführung

Unity ist eine Entwicklungsumgebung zur Erstellung und Gestaltung von Computerspielen. Unity-Projekte können darüber hinaus auch zu Lern- und Trainingszwecken genutzt werden.

Was machen wir mit Unity?

Unity bietet eine Vielfalt von Möglichkeiten. In diesem Einsteigerbuch beschäftigen wir uns mit den wichtigsten Elementen, die Ihnen eine selbstständige Gestaltung und Programmierung Ihrer Projekte ermöglichen.

Ihre Projekte

Wir werden mit einfachen Flächen und Körpern arbeiten, die Ihnen ein Verständnis für Elemente im zweidimensionalen Raum und im dreidimensionalen Raum vermitteln. Daraus bauen wir gemeinsam Schritt für Schritt eine ganze Reihe verschiedener Spiele auf. Dabei lernen wir, wie die Elemente aufeinander reagieren, besonders unter physikalischen Bedingungen.

Physik

Wir lernen die Elemente der Programmierung mit C# von Grund auf kennen, um vielseitige Abläufe gestalten zu können. Dadurch werden Sie in die Lage versetzt, die vorhandenen Spiele nach Ihren Wünschen weiter zu verändern und eigene Spiele selbstständig zu entwickeln.

Abläufe gestalten

Wir werden nicht einfach vorgefertigte komplexe Elemente miteinander kombinieren, wie sie zum Beispiel in großer Zahl im *Asset Store* von Unity angeboten werden. Diese Elemente besitzen zwar eine Fülle von Fähigkeiten und bieten zahlreiche optische Effekte, allerdings trägt das reine Einsetzen und punktuelle Verändern dieser Elemente nur wenig zum Verständnis ihres komplexen Aufbaus bei. Sie vereinfachen auch nicht das Verständnis für den programmierten Spielablauf. Viele reichhaltig gestaltete Spielfiguren können zudem nur mit externen Programmen erstellt werden und müssen danach zunächst in Unity importiert werden.

Verständnis

Die Begriffe *zweidimensional* und *dreidimensional* kürze ich in diesem Buch häufig mit *2D* und *3D* ab. Wir entwickeln zunächst reine 2D-Spiele. Dabei arbeiten wir bereits mit vielen Unity-Elementen, die später auch für die Entwicklung von 3D-Spielen benötigt werden.

2D-Spiele

2D-Spiele finden ausschließlich auf der Ebene des Bildschirms statt. Sie sind übersichtlicher und einfacher zu erfassen als 3D-Spiele. Sie können sich also auf das Erlernen der Spiele-Entwicklung konzentrieren und müs-

sen sich nicht gleichzeitig mit der Orientierung, der Drehung und der Perspektive im dreidimensionalen Raum befassen.

Mein Dank: Für die Hilfe bei der Erstellung dieses Buches bedanke ich mich bei dem ganzen Team vom Rheinwerk Verlag, ganz besonders bei Almut Poll und Joram Seewi.

Wie entsteht der programmierte Spielablauf?

C# Zur Programmierung der logischen Abläufe in den Unity-Projekten können Sie verschiedene Programmiersprachen einsetzen. In diesem Buch wird ausschließlich mit der Programmiersprache C# (sprich: C-Sharp) gearbeitet. Alle benötigten Elemente von C# lernen Sie im Verlauf der Projekte an der passenden Stelle kennen.

Als Programmierumgebung können Sie MonoDevelop oder das Visual Studio von Microsoft nutzen. MonoDevelop ist bereits in Unity integriert. Das Visual Studio müsste separat installiert und mit Unity verbunden werden. In diesem Buch wird ausschließlich mit MonoDevelop gearbeitet. Alle Erläuterungen zur Arbeit mit der Programmierumgebung beziehen sich daher nur auf MonoDevelop.

Programmierkurs In Kapitel 18 finden Sie zusätzlich einen C#-Programmierkurs, in dem ohne die eigentlichen Unity-Elemente gearbeitet wird. Im Vordergrund stehen diejenigen Elemente der Sprache, die Sie in den Unity-Projekten besonders benötigen. Sie können mit dem Programmierkurs beginnen. Oder Sie nutzen ihn als Ergänzung für das bessere Verständnis und als Nachschlagewerk, falls Sie etwas über bestimmte Elemente von C# erfahren möchten.

Unity installieren

Die Anwendung Unity steht Ihnen in zwei Editionen zur Verfügung:

- Professional Edition ▶ In der *Professional Edition*: Sie bietet alle Möglichkeiten zur Entwicklung und ist über verschiedene Lizenzmodelle zu erwerben.
- Personal Edition ▶ In der *Personal Edition*: Sie dürfen sie benutzen, falls Sie beziehungsweise Ihr Unternehmen einen Umsatz von weniger als 100.000 US\$ pro Jahr haben. Die Personal Edition besitzt gegenüber der Professional Edition einige Einschränkungen. Diese treten aber bei einem Einstieg in das Thema noch nicht hervor. Alle Beispiele dieses Buches sind mithilfe der Personal Edition entstanden.

Sie können die Personal Edition von der Unity-Website auf Ihrem Rechner installieren. Sie können dabei wählen, ob Sie die benötigten Dateien während der Installation zusätzlich auf Ihrer Festplatte speichern.

Speichern

In diesem Buch wird mit der Version 2017.3 von Unity unter *Windows 10* gearbeitet, die am 19.12.2017 erschienen ist. In Kapitel 21 wird Unity unter *macOS High Sierra* installiert, ebenfalls in der Version 2017.3. Zudem werden dort mithilfe von Unity unter Windows 10 fertige Anwendungen für verschiedene Plattformen erstellt, zum Beispiel:

Windows, macOS

- ▶ für das Betriebssystem *Android* für die Anwendung auf einem Tablet oder Smartphone oder
- ▶ für die Grafikschnittstelle *WebGL*. Damit können die Anwendungen in allen modernen Browsern unter verschiedenen Betriebssystemen laufen.

Android

WebGL

Dateiendungen anzeigen lassen

Es ist sowohl für die Installation als auch für die Projektentwicklung nützlich, Windows so einzustellen, dass die Endungen der Dateinamen angezeigt werden. Im Explorer von Windows 10 geht das wie folgt:

Windows Explorer

- ▶ Klappen Sie die Liste OPTIONEN auf der Registerkarte ANSICHT auf, und wählen Sie ORDNER- UND SUCHOPTIONEN ÄNDERN.
- ▶ Wechseln Sie im Dialogfeld ORDNEROPTIONEN auf die Registerkarte ANSICHT.
- ▶ Entfernen Sie die Markierung bei ERWEITERUNGEN BEI BEKANNTEN DATEITYPEN AUSBLENDEN.
- ▶ Bestätigen Sie Ihre Änderungen über die Schaltfläche OK.

Durchführung der Installation

Falls Sie Unity von der Unity-Website installieren möchten, suchen Sie am besten die folgende Adresse auf:

<https://unity3d.com/get-unity/download?ref=personal>

In Abbildung 1 sehen Sie die Downloadseite, als Beispiel mit der Version 2017.3. Es können Versionen für die Plattformen Windows und Mac OS X installiert werden. Im unteren Bereich sehen Sie einen Hinweis auf die Beta-Versionen. Die nachfolgende Anleitung kann darauf ebenso angewendet werden.

Unity-Website

Assistent Mit einem Klick wird ein kleines Installationsprogramm von 636 KB heruntergeladen. Für Windows und die genannte Version lautet der Name des Programms *UnityDownloadAssistant-2017.3.0f3.exe*. Es führt uns durch die eigentliche Installation.



Abbildung 1 Herunterladen des Installationsprogramms

Nach dem Aufruf des heruntergeladenen Installationsprogramms müssen Sie zunächst die Lizenzvereinbarung akzeptieren.

Komponenten Es erscheint eine Liste mit Komponenten. Ich empfehle Ihnen, die Komponenten aus Abbildung 2 zu installieren, zusätzlich die Komponente WebGL BUILD SUPPORT, die Sie ganz unten in der Liste finden. Sie können später das Installationsprogramm zum Nachinstallieren weiterer Komponenten erneut aufrufen.

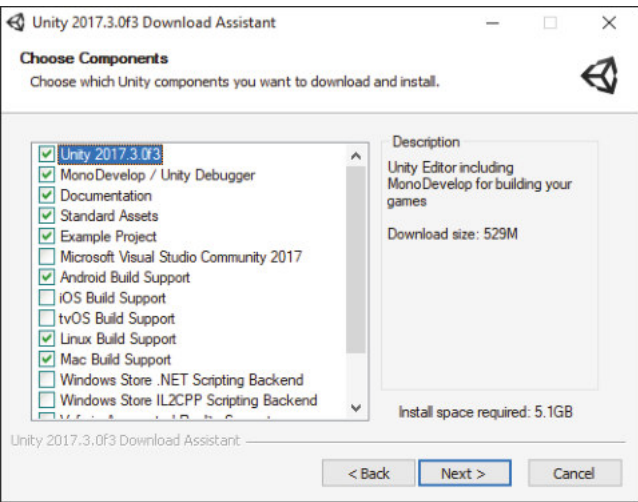


Abbildung 2 Ausgewählte Komponenten

Falls Sie die Installationsdateien der ausgewählten Komponenten dauerhaft speichern möchten, sollten Sie die Option DOWNLOAD TO: und einen geeigneten Ort auswählen (siehe Abbildung 3). Damit haben Sie die Möglichkeit, bei Bedarf eine erneute Installation ohne eine Wiederholung des Downloads durchzuführen.

Dateien speichern

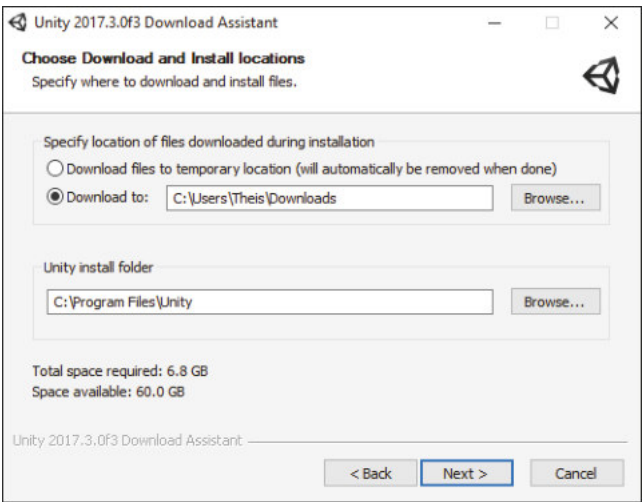


Abbildung 3 Installationsdateien dauerhaft speichern

Nach Betätigung der Schaltfläche NEXT startet die Installation. Die Dateien werden der Reihe nach heruntergeladen und installiert.

Unity wird standardmäßig im Verzeichnis *C:\Programme\Unity* installiert. Nach der Installation wird automatisch eine Verknüpfung zur Anwendung UNITY auf dem Desktop eingerichtet.

Offline-Dokumentation

Nach einer Standardinstallation im Verzeichnis *C:/Programme/Unity* finden Sie in zwei verschiedenen Unterverzeichnissen jeweils eine Datei mit dem Namen *index.html*, und zwar

- ▶ in *Editor/Data/Documentation/en/Manual* und
- ▶ in *Editor/Data/Documentation/en/ScriptReference*.

Legen Sie Verknüpfungen zu diesen beiden Dateien auf Ihren Desktop. Anschließend haben Sie schnell Zugriff auf diese beiden wichtigen Teile der englischsprachigen Offline-Dokumentation.

Manual und Reference Im *Manual* werden mit Beispielen und Bildern die einzelnen Elemente erläutert, aus denen sich Unity-Projekte zusammensetzen. In der *Scripting Reference* oder auch *Scripting API* werden die Klassen beschrieben, die Sie zur Programmierung benötigen.

Unity-Account und Asset Store

Starten Sie Unity nach der Installation über die Verknüpfung auf dem Desktop. Beim allerersten Start werden Sie aufgefordert, einen eigenen *Unity-Account* einzurichten.

Asset Store Ein solcher Account erlaubt Ihnen innerhalb der Entwicklungsumgebung den Zugriff auf den *Asset Store*. Darin finden Sie eine große Auswahl an *Assets*. Dabei handelt es sich um Elemente, die Sie in Ihren Projekten einsetzen können. Es gibt sowohl kostenfreie als auch kostenpflichtige *Assets*. Innerhalb eines Unity-Projekts erhalten Sie bei Bedarf über den Menüpunkt **WINDOW • ASSET STORE** den Zugriff auf den Asset Store. Zur besseren Übersicht ist der Store durchsuchbar und filterbar. Die ausgewählten *Assets* können zudem sortiert werden.

New, Open Nach dem erfolgreichen Einrichten eines Unity-Accounts erscheint beim Start von Unity nunmehr ein Dialogfeld, in dem Sie unter bereits vorhandenen Projekten auswählen können. Über den Link **NEW** können Sie ein neues Projekt anlegen. Über den Link **OPEN** gelangen Sie zu einem Dialogfeld zur Auswahl eines Projekts im Verzeichnis der Unity-Projekte.

Projektverzeichnis und Vorlagen

Beispiele downloaden In diesem Buch arbeite ich mit einer ganzen Reihe von Beispielprojekten, die Ihnen sowohl zur Entwicklung als auch in ausführbarer Form zur Verfügung stehen. Sie können sie sowohl für Windows als auch für macOS High Sierra über die Webseite <https://www.rheinwerk-verlag.de/4654> aus dem Downloadbereich zum Buch herunterladen.

Bei Unity wird jedes Projekt in einem eigenen Projektverzeichnis gespeichert. Legen Sie in Ihrem Windows-Verzeichnis *Dokumente* ein eigenes Unterverzeichnis an, zum Beispiel *UnityProjekte*. Kopieren Sie alle Verzeichnisse mit meinen Beispielprojekten nach dem Download in dieses Verzeichnis. Bei Erstellung eines eigenen Projekts sollten Sie darauf achten, ebenfalls dieses Verzeichnis zu nutzen.

In meinen Beispielprojekten nutze ich zudem eine Reihe von einfachen vorgefertigten Elementen, sogenannte *Assets*. Sie finden sie ebenfalls im Downloadbereich zum Buch, und zwar im Verzeichnis *FreieAssets*. Kopieren Sie dieses Verzeichnis an eine leicht zugängliche Stelle, damit Sie in den einzelnen Abschnitten darauf zugreifen können.

Alle drei bis vier Monate erscheint eine neue Version von Unity. Bezüglich der Beispielprojekte zu diesem Einsteigerbuch erwarte ich aufgrund meiner Erfahrungen mit den letzten Versionen keine gravierenden Veränderungen. Mit diesem Buch sind Sie auch für zukünftige Versionen gerüstet.

Zum Erscheinen der genannten Versionen werden Sie alle Beispielprojekte auch in einer dazu passenden Version auf der Webseite zum Buch finden, und zwar unter <https://www.rheinwerk-verlag.de/4654> bei den **MATERIALIEN ZUM BUCH**.

Dort finden Sie auch zwei neue 2D-Bonusprojekte (siehe Abschnitt 21.4). Sie sind den bekannten Spielen *Pacman* und *Frogger* nachempfunden. Mit den Fähigkeiten, die Sie aus diesem Buch gewonnen haben, sind Sie in der Lage, den Aufbau dieser Projekte zu erkennen und sie mit eigenen Ideen selbstständig weiterzuentwickeln.

Projekt re-importieren

Es ist kein Problem, ein Unity-Projekt zu öffnen, das unter einer der vorherigen Unity-Versionen erstellt wurde. Es wird Ihnen automatisch ein *Re-Import* vorgeschlagen (siehe Abbildung 4). Sie müssen nur die Schaltfläche **CONTINUE** betätigen. Anschließend können Sie das Projekt bearbeiten, verändern und speichern.

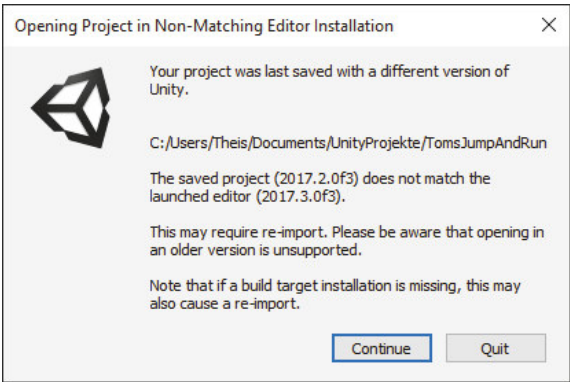


Abbildung 4 Re-Import aus vorheriger Version

Endgültig Bei einem erneuten Öffnen desselben Projekts wird allerdings wiederum ein Re-Import vorgeschlagen. Falls Sie das Projekt endgültig in die aktuelle Unity-Version überführen möchten, sollten Sie zuvor eine ausführbare Version des Unity-Projekts erstellen. Mehr dazu in Abschnitt 3.5.

Kapitel 1

Das erste 2D-Projekt

In diesem Abschnitt lernen wir den UNITY EDITOR als unsere Benutzeroberfläche zur Erstellung von Projekten kennen. Wir legen dazu ein erstes 2D-Projekt mit dem Namen *ZweiDimensionen* an, das einige einfache Elemente beinhaltet.

1.1 Erstellung eines neuen Projekts

Starten Sie Unity. Es erscheint das Dialogfeld zur Auswahl beziehungsweise Erstellung eines Projekts. Betätigen Sie den Link **NEW** zur Erstellung eines neuen Projekts. Im oberen Eingabefeld geben wir den Namen des Projekts ein. In unserem Fall ist das »ZweiDimensionen« (siehe Abbildung 1.1).

New

Das Verzeichnis *UnityProjekte* für die Projektverzeichnisse haben wir bereits in der Einführung im Abschnitt »Projektverzeichnis und Vorlagen« unterhalb des Windows-Verzeichnisses *Dokumente* angelegt. Es sollte zur Erstellung ausgewählt sein.

Es wird die Option 2D markiert, damit passende Voreinstellungen für 2D-Projekte getroffen werden. Sie können zudem entscheiden, ob Projektdaten zu Analysezwecken gesammelt werden dürfen. Nach Betätigung der Schaltfläche **CREATE PROJECT** wird das Projekt erzeugt und im UNITY EDITOR angezeigt.

Create project

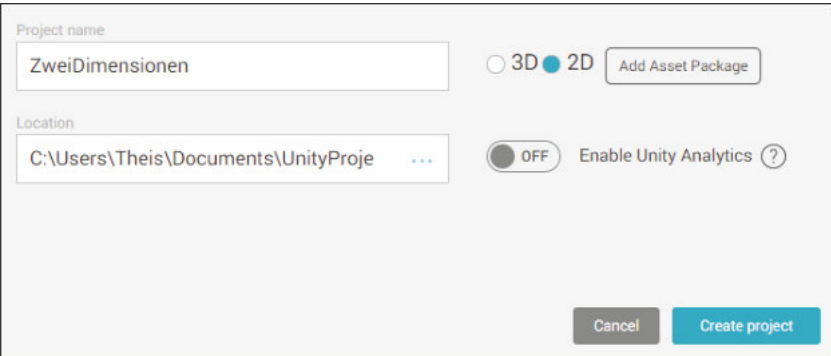


Abbildung 1.1 Erstellung des ersten 2D-Projekts

1.2 Wichtige Bereiche im Unity Editor

Entwicklungs-
umgebung

Der UNITY EDITOR stellt die Umgebung dar, in der wir unsere Unity-Projekte entwickeln. Wir arbeiten mit der englischsprachigen Version von Unity. Ich verwende daher die englischen Begriffe für die Elemente des Programms, wie sie auch vor Ihnen auf dem Bildschirm erscheinen.

Neben der Menüleiste am oberen Rand gibt es noch andere wichtige Bereiche im UNITY EDITOR, die sogenannten Views (siehe Abbildung 1.2). Für jeden Bereich gibt es einen Reiter, auf dem die Bezeichnung der View steht:

Project View

Am unteren Rand finden Sie die PROJECT VIEW. Sie beinhaltet die sogenannten Assets. Assets sind Elemente, die uns für ein Unity-Projekt grundsätzlich zur Verfügung stehen. Wir können einzelne oder auch alle Assets als Spielobjekte oder Teile von Spielobjekten verwenden. Ein Asset kann mehrfach in einem Spiel eingesetzt werden oder eventuell zunächst auch gar nicht. Zudem können Sie in der PROJECT VIEW das Menü CREATE aufklappen, zum Hinzufügen von Assets.

Console View

Ebenfalls am unteren Rand gibt es die CONSOLE VIEW. Sie kann durch einen Klick auf ihren Reiter nach vorne gebracht werden. Hier sehen Sie Kontrollausgaben oder auch Fehlermeldungen Ihrer Programme.

Hierarchy View

Am linken Rand gibt es die HIERARCHY VIEW. Sie dient als ein hierarchisches Verzeichnis derjenigen Spielobjekte, die tatsächlich im Projekt agieren. Spielobjekte können aus den Assets stammen, aber auch auf andere Weise erzeugt werden. Zudem können Sie in der HIERARCHY VIEW das Menü CREATE aufklappen, zum Hinzufügen von Spielobjekten.

Scene View

In der Mitte finden Sie die SCENE VIEW. Sollte sie zunächst hinter der GAME VIEW stehen, kann sie durch einen Klick auf ihren Reiter nach vorne gebracht werden. Hier positionieren Sie während der Entwicklung diejenigen Spielobjekte, die tatsächlich im Projekt agieren, auf dem Spielfeld.

Game View

Ebenfalls in der Mitte gibt es die GAME VIEW. Hier sehen Sie eine Vorschau auf das Spielfeld, wie es während des Spiels aussieht.

Inspector View

Am rechten Rand sehen Sie die INSPECTOR VIEW. Sie beinhaltet die Komponenten desjenigen Spielobjekts, das Sie aktuell in der HIERARCHY VIEW oder in der SCENE VIEW ausgewählt haben. In den Komponenten finden Sie die einzelnen Eigenschaften der Spielobjekte und deren aktuelle Werte. Sie können diese Werte betrachten und verändern.

Asset Store

Sollte neben der SCENE VIEW und der GAME VIEW auch der Asset Store erscheinen, können Sie ihn schließen. Klicken Sie dazu mit der rechten

Maustaste auf den Reiter, und wählen Sie im Kontextmenü den Menüpunkt CLOSE TAB.

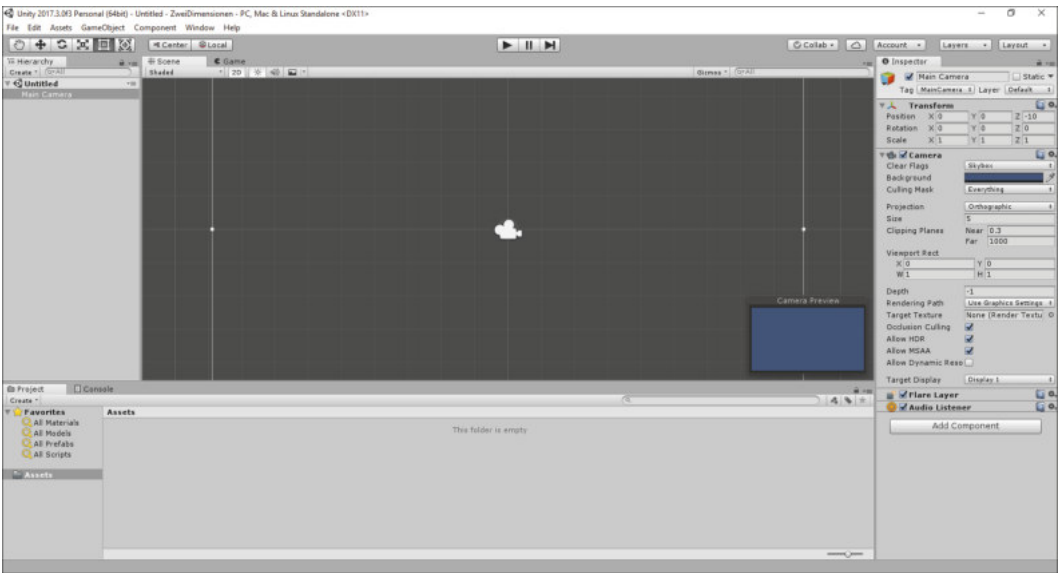


Abbildung 1.2 Der »Unity Editor«

Alle Bereiche können in ihrer Größe verändert werden, indem sie am Rand mithilfe der Maus angefasst werden. An ihrem Reiter können Sie ebenfalls mithilfe der Maus angefasst und an andere Stellen des Bildschirms verschoben werden. Sie können Bereiche ganz ausblenden oder weitere Bereiche einblenden.

Reiter

Die Veränderungen über den Reiter sollten zunächst mit Vorsicht durchgeführt werden. Ganz oben rechts gibt es eine Liste von möglichen Layouts. Jedes Layout bietet typische Werte für die Anordnung und die Größe der Bereiche (siehe Abbildung 1.3). Über den Eintrag Default können Sie die Grundeinstellung wieder aufrufen.

Layouts

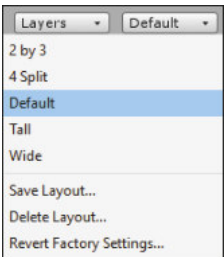


Abbildung 1.3 Liste der Layouts

1.3 Das Spielobjekt »Main Camera«

- Sicht des Benutzers
- Jedes 2D-Projekt umfasst standardmäßig bereits ein Spielobjekt, das in der HIERARCHY VIEW erscheint. Es hat den Namen *Main Camera* und stellt die Kamera dar, mit deren Hilfe der Benutzer auf das Spielfeld sieht. Es erscheint in der Mitte der SCENE VIEW in Form eines Kamerasymbols. In der HIERARCHY VIEW können Sie die Sichtbarkeit der Spielobjekte mithilfe des Pfeils links neben ihrem Namen einstellen.
- Eigenschaften
- Falls Sie wie in Abbildung 1.2 das Spielobjekt *Main Camera* in der HIERARCHY VIEW markiert haben, sehen Sie rechts unten in der SCENE VIEW das kleine Fenster *CAMERA PREVIEW*, das eine Vorausschau auf das spätere Spielfeld ermöglicht. Gleichzeitig sehen Sie in der INSPECTOR VIEW die Eigenschaften und Werte für das Spielobjekt *Main Camera*. Auf einige dieser Werte gehe ich später noch ein.

1.4 Assets importieren

Auf Basis eines neuen Assets sollen zwei Spielobjekte eingefügt werden. Gemeinsam mit den Beispielprojekten haben Sie in der Einführung im Abschnitt »Projektverzeichnis und Vorlagen« das Verzeichnis *FreieAssets* heruntergeladen und an einer leicht zugänglichen Stelle gespeichert. Darin befinden sich einige einfache Bilddateien, die als Assets importiert werden können. Eine davon benötigen wir.

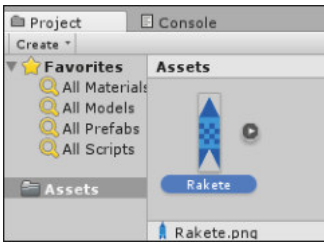


Abbildung 1.4 Asset aus Bilddatei

- Freie Assets
- Rufen Sie den Menüpunkt **ASSETS • IMPORT NEW ASSET** auf. Wählen Sie die Bilddatei *Rakete.png* aus dem Verzeichnis *FreieAssets* aus, und betätigen Sie die Schaltfläche **IMPORT**. Nach dem Import erscheint Ihr erstes Asset mit dem Namen *Rakete* in der PROJECT VIEW im Bereich **ASSETS**. Es handelt sich um ein Asset des Typs *Sprite*. Sprites sind einfache zweidimensionale Grafikobjekte. Falls Sie es in der PROJECT VIEW markieren, wird unten der

Dateiname angezeigt (siehe Abbildung 1.4). Über den Schieberegler ganz rechts unten in der PROJECT VIEW können Sie die Größe der Darstellung der Assets verändern.

1.5 Spielobjekte einfügen

Ziehen Sie das Asset zweimal nacheinander mithilfe der Maus in die HIERARCHY VIEW. Achten Sie dabei darauf, das Asset nicht genau auf einem bereits vorhandenen Spielobjekt in der HIERARCHY VIEW loszulassen, sondern in dem freien Bereich unterhalb der bisher vorhandenen Spielobjekte. Damit haben Sie zusätzlich zum Spielobjekt *Main Camera* zwei weitere Spielobjekte in das Projekt eingefügt. Diese tragen zunächst die Namen *Rakete* und *Rakete (1)* und entstammen demselben Asset.

Markieren Sie das erste neue Spielobjekt *Rakete* in der HIERARCHY VIEW. Nun können Sie seine Eigenschaften betrachten oder ändern. Wenn Sie es noch einmal anklicken, können Sie seinen Namen ändern, zum Beispiel in *RaketeEins*.

In der INSPECTOR VIEW sehen Sie in der Komponente **SPRITE RENDERER** die Eigenschaft **MATERIAL**. Ihr Wert sollte auf *Sprites/Default* stehen (siehe Abbildung 1.5), damit die Bilddatei unverfälscht dargestellt wird. Falls das einmal nicht der Fall sein sollte, klicken Sie auf den Kreis rechts neben der Eigenschaft **MATERIAL**. Es öffnet sich das Dialogfeld **SELECT MATERIAL**. Wählen Sie darin den Eintrag *Sprites/Default* aus.

Erste Spielobjekte

Umbenennen

Sprites/Default

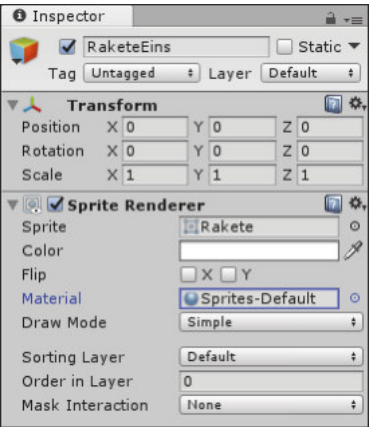


Abbildung 1.5 Komponente »Sprite Renderer«, Eigenschaft »Material«

Führen Sie die beiden Aktionen auch für das zweite neue Spielobjekt Rakete (1) aus. Es soll RaketeZwei heißen (siehe Abbildung 1.6). Beide Raketen liegen in der SCENE VIEW noch genau hintereinander. Das wird in Kürze geändert.

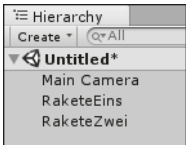


Abbildung 1.6 Zwei neue Spielobjekte

1.6 Ändern der Hierarchie

In diesem Abschnitt zeige ich Ihnen einige typische Änderungen in der HIERARCHY VIEW, die erfahrungsgemäß auch schnell zu Fehlern führen können. Führen Sie die Änderungen auch selber einmal durch. Solange mögliche Fehler in diesem kleinen Übungsprojekt auftreten, sind die Auswirkungen noch überschaubar.

Reihenfolge der Spielobjekte

Aus Gründen der Übersichtlichkeit ist die Reihenfolge der Spielobjekte in der HIERARCHY VIEW wichtig. Nehmen wir an, Sie möchten das Spielobjekt RaketeZwei vor dem Spielobjekt RaketeEins anordnen. Dazu betätigen Sie auf dem Spielobjekt RaketeZwei die linke Maustaste, halten sie gedrückt und ziehen das Spielobjekt nach oben, bis der Mauszeiger als eine dünne blaue Linie zwischen den Spielobjekten Main Camera und RaketeEins erscheint. In diesem Moment können Sie die Maustaste loslassen.

Rückgängig

Die Spielobjekte haben nun die gewünschte Reihenfolge. Mit dem Menübefehl EDIT / UNDO ... beziehungsweise der Tastenkombination `[Strg]+[Z]` machen Sie diese Änderung wieder rückgängig.

Unterordnung

Falls Sie das Spielobjekt RaketeZwei bei der oben genannten Änderung genau auf dem Spielobjekt RaketeEins loslassen (siehe Abbildung 1.7), wird es dem Spielobjekt RaketeEins hierarchisch untergeordnet (siehe Abbildung 1.8).



Abbildung 1.7 Falscher Zeitpunkt zum Loslassen

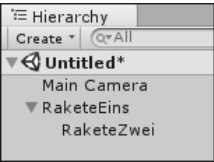


Abbildung 1.8 Falsche Hierarchie von Spielobjekten

Diese Unterordnung kann aus zwei Gründen erwünscht sein:

- Sie haben ein Spielobjekt, das aus mehreren untergeordneten Spielobjekten zusammengesetzt wird. Das kann zum Beispiel das Spielobjekt Mensch sein, das aus den Körperteilen (sprich: *Spielobjekten*) Kopf, Rumpf, Linker Arm, Rechter Arm, Linkes Bein und Rechtes Bein besteht. Falls der Mensch als Ganzes bewegt wird, bewegen sich alle seine Körperteile mit ihm. Es gibt also das übergeordnete Spielobjekt Mensch und sechs untergeordnete Spielobjekte für die einzelnen Körperteile (siehe Abbildung 1.9).
- Sie haben sehr viele Spielobjekte, die Sie thematisch zusammenfassen möchten, damit Sie in der HIERARCHY VIEW nicht den Überblick verlieren. Dabei dient das übergeordnete Spielobjekt als *Verzeichnis*, vergleichbar einem Verzeichnis im Dateisystem Ihres Rechners. Das kann zum Beispiel das Verzeichnis Raum sein, in dem sich die Spielobjekte Boden, Decke, Linke Wand und Rechte Wand befinden (siehe ebenfalls Abbildung 1.9).

Gemeinsam bewegen

Verzeichnis

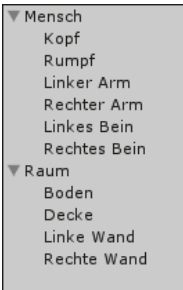


Abbildung 1.9 Spielobjekte als übergeordnete Verzeichnisse

Ein übergeordnetes Spielobjekt wird *Parent* (dt.: Elternteil) genannt, ein untergeordnetes Spielobjekt *Child* (dt.: Kind). Die Mehrzahl von Child ist *Children* (dt.: Kinder). Der Vorgang des Unterordnens innerhalb einer Unity-Hierarchie wird *Parenting* genannt. Alle diese Begriffe werden uns in den Unity-Projekten begegnen.

Parent und Child

In diesem Projekt ist die Unterordnung allerdings nicht erwünscht. Machen Sie diese Änderung daher mithilfe der Tastenkombination `Strg`+`Z` rückgängig, sodass sich wieder die ursprüngliche Reihenfolge wie in Abbildung 1.6 ergibt.

1.7 Eine Szene speichern

Save Scenes Sie sollten eine Spielszene nach dem Einfügen von Spielobjekten oder nach anderen Änderungen regelmäßig speichern. Rufen Sie dazu den Menüpunkt `FILE • SAVE SCENES` auf. Es wird das Unterverzeichnis `Assets` Ihres Projekts zur Speicherung angeboten. Speichern Sie die Szene dort, zum Beispiel unter dem Namen `Szene1`. Szene-Dateien haben die Endung `unity`, wie Sie in der `PROJECT VIEW` in Abbildung 1.10 sehen.

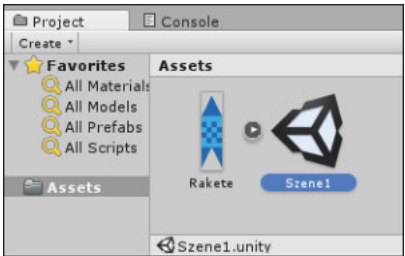


Abbildung 1.10 Asset des Typs »Szene«

Später genügt die Tastenkombination `Strg`+`S` zum erneuten Speichern aller vorhandenen Szenen. Ein Projekt kann aus mehreren Szenen bestehen, die nacheinander durchlaufen werden. Unser Projekt hat nur eine Szene.

1.8 Die Komponente »Transform«

Position, Drehung und Größe Jedes Spielobjekt besitzt die Komponente `TRANSFORM`. Dabei handelt es sich um die wichtigste Komponente. Sie dient zur Festlegung der Position, der Drehung und der Größe eines Spielobjekts. Markieren Sie einmal das Spielobjekt `RaketeEins` in der `HIERARCHY VIEW` oder in der `SCENE VIEW`. In der `INSPECTOR VIEW` sehen Sie nun seine Komponenten, wie zum Beispiel die Transform-Komponente in Abbildung 1.11.

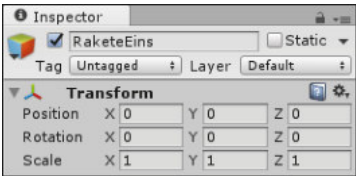


Abbildung 1.11 Spielobjekt »RaketeEins«, Transform-Komponente

1.8.1 Die Eigenschaften der Transform-Komponente

In der Transform-Komponente können Sie die nachfolgenden Eigenschaften eines Spielobjekts sehen und verändern.

Die Eigenschaft `POSITION`:

Sie gibt die Position des Spielobjekts auf dem Spielfeld entlang der drei Koordinatenachsen an. Die X-Achse verläuft von links nach rechts. Ein Spielobjekt hat am linken Rand des Spielfelds also zum Beispiel den X-Wert -5, in der Mitte den X-Wert 0 und am rechten Rand den X-Wert 5. Die Y-Achse verläuft von unten nach oben. Ein Spielobjekt hat am unteren Rand des Spielfelds also zum Beispiel den Y-Wert -5, in der Mitte den Y-Wert 0 und am oberen Rand den Y-Wert 5.

X- und Y-Achse

Bei 2D-Spielen wird die Z-Koordinate für die dritte Dimension nur selten benötigt. Die Z-Achse verläuft von vorne (vom Benutzer aus *vor* dem Bildschirm) nach hinten (vom Benutzer aus *hinter* dem Bildschirm). Ein Spielobjekt hat vor dem Bildschirm also zum Beispiel den Z-Wert -5, auf der Ebene des Bildschirms den Z-Wert 0 und hinter dem Bildschirm den Z-Wert 5.

Z-Achse

Zur Verdeutlichung der drei Achsen dient die *Linke-Hand-Regel*:

Linke-Hand-Regel

- Der Daumen der linken Hand soll nach rechts weisen. Damit zeigt er in Richtung der positiven X-Achse.
- Der Zeigefinger der linken Hand soll nach oben weisen. Damit zeigt er in Richtung der positiven Y-Achse.
- Der Mittelfinger der linken Hand soll abgewinkelt werden und nach hinten weisen, sozusagen durch den Bildschirm hindurch. Damit zeigt er in Richtung der positiven Z-Achse.

Das Spielobjekt `Main Camera` stellt bezüglich der Z-Koordinate eine Ausnahme dar. Mit einem Wert von -10 steht es vor dem Bildschirm. Daher schauen wir von vorne auf das Spielfeld.

Die Eigenschaft ROTATION:

Drehung in Grad Sie gibt die Drehung des Spielobjekts um die X-Achse, um die Y-Achse und um die Z-Achse als Winkel in Grad an. In 2D-Projekten ist normalerweise nur eine Drehung um die Z-Achse, also eine Drehung in der Ebene des Bildschirms, sinnvoll. Ein positiver Wert bedeutet eine Drehung um die Z-Achse gegen den Uhrzeigersinn.

Die Eigenschaft SCALE:

Faktor für Größe Sie gibt die relative Größe des Spielobjekts in der X-Richtung, in der Y-Richtung und in der Z-Richtung mithilfe eines Skalierungsfaktors an. Zunächst haben alle drei Faktoren den Wert 1.

1.8.2 Werte in der »Inspector View« ändern

Nach dem Einfügen eines neuen Spielobjekts besitzt es zunächst die folgenden Transform-Werte (siehe auch Abbildung 1.11):

- ▶ Es liegt in der Mitte des Spielfelds, bei X = 0, Y = 0 und Z = 0, oder – verkürzt ausgedrückt – an der Position 0/0/0.
- ▶ Es ist nicht gedreht, die Rotation liegt also ebenfalls bei 0/0/0.
- ▶ Es hat seine Originalgröße, der Scale steht also bei 1/1/1.

In Abbildung 1.12 sehen Sie das Spielfeld, nachdem die Transform-Werte für die Spielobjekte RaketeEins und RaketeZwei in der INSPECTOR VIEW geändert wurden. Die Einstellung der Kamera wurde nicht geändert.

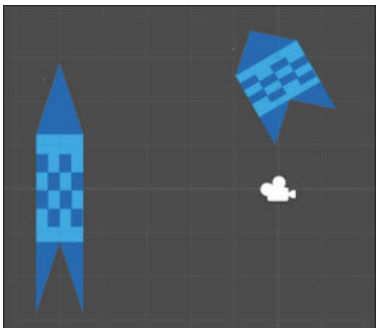


Abbildung 1.12 Transform-Werte geändert

Ort und Größe geändert Links in Abbildung 1.12 sehen Sie das Spielobjekt RaketeEins an der Position -5/0/0. Es ist nicht gedreht, die Rotation beträgt also 0/0/0. Es ist in X-

Richtung leicht und in Y-Richtung etwas stärker vergrößert. Die Größe in Z-Richtung ist nicht relevant und kann auf 0 gesetzt werden. Der Scale beträgt nun 1.2/1.8/0.

Die Stellen nach dem Komma werden, wie im englischen Sprachraum üblich, mit einem *Dezimalpunkt* statt einem *Dezimalkomma* abgetrennt. So sehen Sie es auch in den Abbildungen dieses Buches, und so müssen Sie auch später die Werte im Programmcode eingeben. Zur Verminderung der Fehlerquote nutze ich daher auch im erläuternden Text dieses Buches den im deutschen Sprachraum eigentlich unüblichen Dezimalpunkt. Es gibt eine kleine Hilfestellung: Falls Sie in der INSPECTOR VIEW einen Wert mit einem Dezimalkomma eingeben, wird daraus automatisch ein Dezimalpunkt.

Oben rechts in Abbildung 1.12 sehen Sie das Spielobjekt RaketeZwei an der Position 0/2.5/0. Es ist um 30 Grad gegen den Uhrzeigersinn um die Z-Achse gedreht, die Rotation beträgt also 0/0/30. Es ist in Y-Richtung leicht verkleinert und in X-Richtung etwas vergrößert, der Scale beträgt 1.8/0.8/0.

Falls Sie bei einer der Rotationsachsen einen Winkel zwischen 180 und 360 Grad einstellen, kann es vorkommen, dass Unity diesen Wert automatisch auf den zugehörigen negativen Wert ändert. Das ist kein Problem und hat keinen Einfluss auf den Ablauf. Ein Beispiel: Der Wert 330 Grad wird geändert auf -30 Grad, also 330 Grad minus 360 Grad.

Ändern Sie in der INSPECTOR VIEW einzelne Transform-Werte der beiden Spielobjekte, und schauen Sie sich die Auswirkungen unmittelbar nach der Änderung an. Damit lernen Sie den Umgang mit den Koordinaten, Achsen, Drehungen und Größenverhältnissen recht schnell.

Alternativ können Sie die Transform-Werte auch mithilfe der Maus ändern. Das wird im folgenden Abschnitt 1.9 beschrieben.

1.9 Die Ansicht in der »Scene View«

Sie haben mehrere Möglichkeiten, die Ansicht in der SCENE VIEW zu ändern. Drücken Sie zunächst einmal die Taste **[Alt]**, und halten Sie sie gedrückt. Der Mauszeiger ändert sich über der SCENE VIEW in eine Hand. Nutzen Sie nun zusätzlich eine der folgenden Maustasten:

- Verschieben

- Verschieben Sie mithilfe der gedrückten linken Maustaste die Ansicht in der SCENE VIEW nach links, rechts, oben oder unten.
- Zoomen

- Zoomen Sie in die SCENE VIEW hinein, und zwar mit der gedrückten rechten Maustaste und einer Verschiebung der Maus nach rechts oder unten.
 - Zoomen Sie aus der SCENE VIEW heraus, und zwar mit der gedrückten rechten Maustaste und einer Verschiebung der Maus nach links oder oben.
- Fokussieren

Zur genauen Betrachtung und Bearbeitung eines einzelnen Spielobjekts können Sie es in der Mitte der SCENE VIEW fokussieren. Markieren Sie es dazu in der HIERARCHY VIEW, setzen Sie die Maus über die SCENE VIEW, und drücken Sie anschließend die Taste **[F]**. Das Spielobjekt wird herangezoomt und rückt in die Mitte der SCENE VIEW. Dasselbe erreichen Sie auch über einen Doppelklick auf das Spielobjekt in der HIERARCHY VIEW. Anschließend können Sie zum Beispiel das Spielobjekt Main Camera fokussieren, um wieder herauszuzoomen.

1.9.1 Positionswerte mithilfe der Maus ändern

- Symbolleisten

Oben links stehen Ihnen zwei Symbolleisten mit insgesamt sieben Schaltflächen zur Verfügung, die unter anderem zur Änderung der Transform-Werte von Spielobjekten mithilfe der Maus dienen (siehe Abbildung 1.13). Ich habe sie zur besseren Übersicht nummeriert.
- Local, Global

Mithilfe der Schaltfläche Nr. 8 können Sie zwischen den Zuständen LOCAL und GLOBAL umschalten:

 - Die Einstellung LOCAL bedeutet, dass Sie in der SCENE VIEW mit den lokalen Achsen arbeiten, die sich auf das Spielobjekt beziehen.
 - Die Einstellung GLOBAL bedeutet, dass Sie in der SCENE VIEW mit den globalen Achsen arbeiten, die sich auf das gesamte Spielfeld beziehen.

Die Bedeutung wird nachfolgend noch klarer. Stellen Sie den Schalter zunächst auf LOCAL.

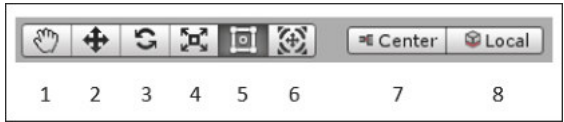


Abbildung 1.13 Zwei Symbolleisten

Betätigen Sie die Schaltfläche Nr. 2, und markieren Sie eines der Spielobjekte, zum Beispiel RaketeZwei. Über dem Spielobjekt erscheint ein *Handle* zur Änderung der Position (siehe Abbildung 1.14).

Fassen Sie mit der Maus an einer der beiden Pfeilspitzen des Handles an, und verschieben Sie das Spielobjekt entlang seiner lokalen X-Achse (rot) oder entlang seiner lokalen Y-Achse (grün). Beachten Sie parallel die Änderung der Positionswerte in der INSPECTOR VIEW.

Stellen Sie den oben genannten Schalter nun auf GLOBAL. Das Handle zur Änderung der Position erscheint mit den globalen X- und Y-Achsen, also den Achsen des gesamten Spielfelds (siehe Abbildung 1.15).

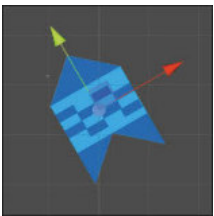


Abbildung 1.14 Position entlang der lokalen Achsen ändern

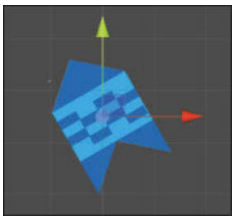


Abbildung 1.15 Position entlang der globalen Achsen ändern

Alternativ können Sie mit der Maus auch das blaue Quadrat anfassen, das sich rechts oberhalb der Mitte des Spielobjekts befindet. Sie können es damit frei auf der X-Y-Ebene verschieben. Das ist die Ebene, die durch die X-Achse und die Y-Achse gebildet wird

1.9.2 Rotationswerte mithilfe der Maus ändern

Stellen Sie den oben genannten Schalter nun wieder auf LOCAL. Betätigen Sie die Schaltfläche Nr. 3 in Abbildung 1.13, und markieren Sie eines der Spielobjekte, zum Beispiel wiederum RaketeZwei. Über dem Spielobjekt erscheint ein Handle zur Änderung der Rotation (siehe Abbildung 1.16).

Fassen Sie mit der Maus an einem der beiden Kreise an, und drehen Sie damit das Spielobjekt um die Z-Achse. Beachten Sie parallel die Änderung der Rotationswerte in der INSPECTOR VIEW.

Falls Sie den oben genannten Schalter wieder auf GLOBAL stellen, ändern sich die Kreise für die Änderung um die Z-Achse nicht. Da bei 2D-Projekten nur Rotationen um die Z-Achse sinnvoll sind, hat hier die Änderung des Schalters keine Auswirkung.

Positions-Handle

Verschieben entlang Achse

Globale Achsen

Verschieben auf Ebene

Rotations-Handle

Drehen um Achse

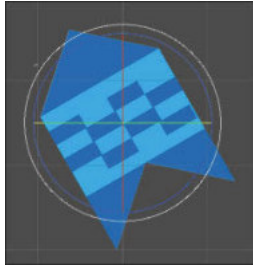


Abbildung 1.16 Rotation ändern

1.9.3 Scale-Werte mithilfe der Maus ändern

Scale-Handle Betätigen Sie die Schaltfläche Nr. 4 in Abbildung 1.13, und markieren Sie eines der Spielobjekte. Über dem Spielobjekt erscheint ein Handle zur Änderung des Scales (siehe Abbildung 1.17):

Einzel ändern ► Fassen Sie mit der Maus an einem der beiden äußeren Quadrate an, und vergrößern oder verkleinern Sie das Spielobjekt in der Höhe oder in der Breite.

Proportional ändern ► Fassen Sie mit der Maus an dem weißen Quadrat in der Mitte an, und vergrößern oder verkleinern Sie das Spielobjekt unter Beibehaltung des Verhältnisses von Höhe zu Breite.

Beachten Sie parallel die Änderung der Scale-Werte, das sogenannte *Skalieren*, in der INSPECTOR VIEW. Sie können über den oben genannten Schalter nicht zwischen GLOBAL oder LOCAL wechseln, da das keine Auswirkung für die Änderung eines Scale-Werts hätte.

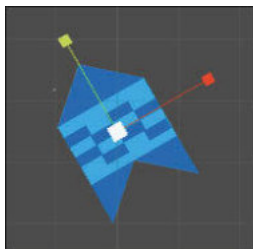


Abbildung 1.17 Scale ändern

Speichern Schließen Sie das Projekt *ZweiDimensionen* und Unity selber über den Menüpunkt FILE • EXIT. Sollten Sie die Szene vorher nicht gespeichert haben, werden Sie nun dazu aufgefordert.

Kapitel 8

Das erste 3D-Projekt

Mit Einführung der dritten Dimension werden Ihre Projekte räumlicher und realistischer. Gleichzeitig werden sie auch komplexer. Nach dem Bearbeiten der bisherigen Kapitel haben Sie aber einen Vorteil: Sie haben bereits viele Unity-Elemente und grundsätzliche Abläufe in 2D-Projekten kennengelernt und können sie in 3D-Projekten in gleicher oder ähnlicher Form weiterverwenden.

Ähnliche Elemente

In einem ersten 3D-Projekt mit dem Namen *DreiDimensionen* werden mithilfe von einfachen 3D-Objekten einige Grundlagen erläutert. Ähnlich wie im Einführungsprojekt *ZweiDimensionen* aus Kapitel 1 handelt es sich nicht um ein Spielprojekt, sondern dient Ihrem Verständnis für typische 3D-Elemente. Einige der Abläufe können Sie zudem in Ihren eigenen 3D-Projekten einsetzen.

Verständnis

8.1 Grundlagen eines 3D-Projekts

Sie erfahren mehr über die Kamera sowie die Elemente und Materialien, aus denen einfache 3D-Objekte zusammengesetzt sind. Zudem gestalten Sie die dreidimensionale Ansicht in der SCENE VIEW.

8.1.1 Kamera, Skybox und Licht

Erstellen Sie ein neues Projekt mit dem Namen *DreiDimensionen*. Achten Sie darauf, dass die Option 3D markiert ist. Damit erhalten 3D-Projekte die passenden Voreinstellungen.

3D-Projekt anlegen

Wie bei 2D-Projekten gibt es zu Beginn bereits das Spielobjekt *Main Camera*. Wir setzen es auf die Position $-2/1/-7$ und schauen damit aus einer leicht erhöhten Position von links vorne auf die Szene. In der Komponente *CAMERA* des Spielobjekts *Main Camera* finden Sie die Eigenschaft *CLEAR FLAGS*. Sie steht auf dem Standardwert *Skybox*. Mit dieser Einstellung wird der Eindruck einer realen Szene inklusive eines Himmels und eines Horizonts erzeugt. Dagegen würde der Hintergrund zum Beispiel mithilfe des Werts *SOLID COLOR* nur in einer einheitlichen Farbe dargestellt.

Horizont

Licht-Objekt Außerdem gibt es in 3D-Projekten zu Beginn ein Licht-Objekt. Es sorgt für eine Beleuchtung der Elemente, zur Erzeugung von Schatten und zur weiteren Verbesserung der räumlichen Wirkung. Sie können in einem Projekt mehrere Licht-Objekte gleichzeitig einsetzen.

Hier handelt es sich um ein Licht-Objekt des Typs *Directional Light*. Wir belassen es bei den Standardwerten, also bei der Position 0/3/0 und bei der Rotation 50/330/0.

Directional Light

Es gibt unter anderem die folgenden Typen von Licht-Objekten:

► Ein Objekt des Typs *Directional Light* beleuchtet die gesamte Szene aus einer bestimmten Richtung. Die Lichtstärke ist überall gleich. Die Wirkung ist ähnlich wie beim Licht der Sonne.

Point Light

► Ein Objekt des Typs *Point Light* strahlt von einem bestimmten Punkt aus in alle Richtungen. Die Lichtstärke nimmt mit der Entfernung zu diesem Punkt ab. Die Wirkung ist ähnlich wie beim Licht einer Glühbirne.

Spot Light

► Ein Objekt des Typs *Spot Light* strahlt ebenfalls von einem bestimmten Punkt aus. Die Lichtstärke nimmt ebenso mit der Entfernung zu diesem Punkt ab. Allerdings wird das Licht mithilfe eines Winkels eingeschränkt, sodass sich ein Lichtkegel ergibt. Die Wirkung in der Szene ist ähnlich wie beim Licht eines Scheinwerfers.

8.1.2 Einfache 3D-Objekte

Sie können in Ihren 3D-Unity-Projekten beliebige komplexe 3D-Objekte einsetzen. Diese 3D-Objekte können mithilfe von verschiedenen Modellierungsprogrammen außerhalb von Unity erstellt werden. Im UNITY EDITOR werden aber bereits einige einfache 3D-Objekte bereitgestellt, die für viele Projekte ausreichen. Einige dieser Objekte werden nachfolgend eingefügt:

Würfel

► Erzeugen Sie einen Würfel mithilfe des Menüpunkts **GAMEOBJECT • 3D OBJECT • CUBE**. Er verbleibt in der Mitte, und zwar an der Position 0/0/0.

Kugel

► Erzeugen Sie als Nächstes mithilfe des Menüpunkts **GAMEOBJECT • 3D OBJECT • SPHERE** eine Kugel. Positionieren Sie sie bei 0/2/0. Sie wird also auf der positiven Y-Achse verschoben und befindet sich oberhalb des Würfels.

Kapsel

► Es folgt eine Kapsel, die mithilfe des Menüpunkts **GAMEOBJECT • 3D OBJECT • CAPSULE** erstellt wird. Sie wird auf der positiven X-Achse verschoben, und zwar an die Position 4/0/0.

► Als Letztes erstellen Sie über den Menüpunkt **GAMEOBJECT • 3D OBJECT • CYLINDER** noch einen Zylinder. Er wird auf der negativen X-Achse an die Position -4/0/0 verschoben.

Zylinder

In der SCENE VIEW ergibt sich ein Bild wie in Abbildung 8.1.

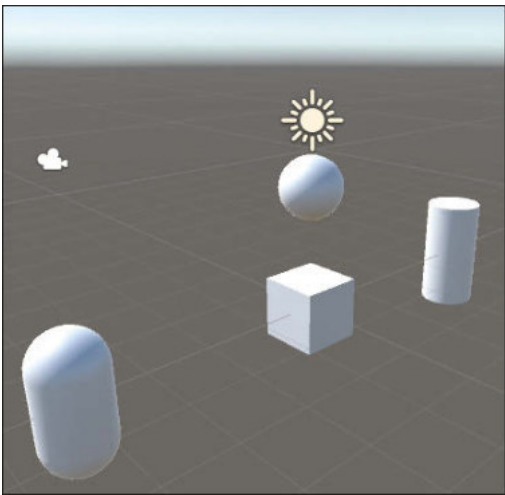


Abbildung 8.1 Einfache 3D-Objekte in der »Scene View«

Der Einfluss des Lichts, der Schattenwurf und die Skybox sind gut zu erkennen. Standardmäßig werden die Objekte perspektivisch dargestellt. Objekte, die weiter weg vom Betrachter sind, werden also kleiner dargestellt. Auch das trägt zur Verbesserung der räumlichen Wirkung bei.

Perspektivische Darstellung

8.1.3 Oberflächenmaterial erstellen und zuordnen

Die Oberflächen der 3D-Objekte sollen eine Farbe erhalten. Erzeugen Sie dazu in der PROJECT VIEW mithilfe des Menüpunkts **ASSETS • CREATE • MATERIAL** ein neues Asset für Oberflächenmaterialien. Nennen Sie es **DBraunMat**, als Abkürzung für *Dunkelbraunes Material*.

Create Material

Markieren Sie das Asset, und stellen Sie in der INSPECTOR VIEW die wichtigste Farbe ein: Wählen Sie für die Eigenschaft **ALBEDO** im Bereich **MAIN MAPS** (siehe Abbildung 8.2) die RGB-Werte 184/124/62. Die Eigenschaft **ALBEDO** bestimmt das Reflexionsverhalten von Oberflächen, die angeleuchtet werden.

Albedo

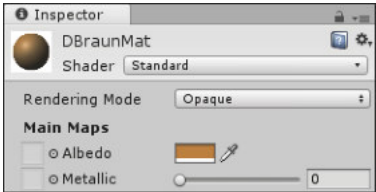


Abbildung 8.2 Wichtigste Farbe für ein Oberflächenmaterial

Material zuordnen

Das Oberflächenmaterial kann allen vier Objekten auf einmal zugeordnet werden. Markieren Sie sie daher gemeinsam in der HIERARCHY VIEW. Klappen Sie in der Komponente MESH RENDERER den Bereich MATERIALS auf. Ziehen Sie das Material-Asset DBraunMat auf die Eigenschaft ELEMENT 0 (siehe Abbildung 8.3).



Abbildung 8.3 Oberflächenmaterial zuordnen

8.1.4 Oberflächenmaterial wechseln

Sie können das Aussehen von 3D-Objekten zur Laufzeit eines Projekts verändern, indem Sie zum Beispiel das Oberflächenmaterial wechseln. Für das vorliegende Codebeispiel wird zuvor das Material-Asset DGruenMat (als Abkürzung für *Dunkelgrünes Material*) mit den RGB-Werten 85/118/57 für die Eigenschaft ALBEDO erzeugt.

Klasse »Sphere«

Es folgt der Code für den Wechsel des Oberflächenmaterials der Kugel:

```
using UnityEngine;
public class Sphere : MonoBehaviour
{
    public Material dBraunMat;
    public Material dGruenMat;

    void Update()
    {
        if (Input.GetKeyDown (KeyCode.B))
            GetComponent<MeshRenderer>().material = dBraunMat;
        else if (Input.GetKeyDown (KeyCode.G))
            GetComponent<MeshRenderer>().material = dGruenMat;
    }
}
```

```
GetComponent<MeshRenderer>().material = dGruenMat;
    }
}
```

Listing 8.1 Klasse »Sphere«

Es werden die beiden öffentlich zugänglichen Variablen dBraunMat und dGruenMat vom Datentyp Material erstellt. Falls der Benutzer die Taste [B] oder die Taste [G] betätigt, erhält die Eigenschaft material der Komponente MESH RENDERER des zugehörigen Spielobjekts einen neuen Wert (siehe Abbildung 8.4).

Das C#-Script wird dem Spielobjekt Sphere zugeordnet. Im UNITY EDITOR werden die beiden Material-Assets auf die zugehörigen Slots gezogen.

Mesh Renderer

Material zuordnen

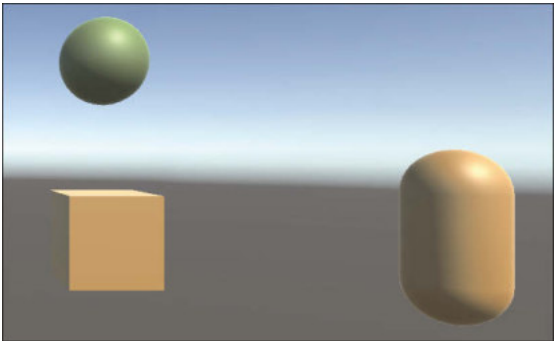


Abbildung 8.4 Nach einem Wechsel des Oberflächenmaterials

8.1.5 Ansicht in der »Scene View« gestalten

Da wir nun einige Objekte zur Verfügung haben, können wir uns unsere Möglichkeiten veranschaulichen, die Ansicht in der SCENE VIEW zu gestalten. Klicken Sie zunächst auf das Hand-Symbol in der oberen linken Symbolleiste:

Hand-Symbol

- Verschieben Sie die Ansicht in der SCENE VIEW durch Verschieben der Maus bei gedrückter linker Maustaste nach links, rechts, oben oder unten.
- Drehen Sie die Ansicht in der SCENE VIEW um verschiedene Achsen durch Drücken und Festhalten der Taste [Strg] und gleichzeitiges Verschieben der Maus bei gedrückter rechter Maustaste nach links, rechts, oben oder unten.

Verschieben

Drehen

Zoomen ► *Zoomen* Sie in der SCENE VIEW durch Drücken und Festhalten der Taste **[Alt]** und gleichzeitiges Verschieben der Maus bei gedrückter rechter Maustaste nach links, rechts, oben oder unten.

Gizmo Rechts oben in der SCENE VIEW befindet sich ein kleines Hilfselement, ein sogenanntes *Gizmo* (siehe Abbildung 8.5). Es zeigt die aktuelle Lage der drei Koordinatenachsen und die aktuelle Darstellungsart.

Linke-Hand-Regel Wie in Abschnitt 1.8.1 wenden Sie auch hier die *Linke-Hand-Regel* an: Der Daumen der linken Hand weist in Richtung der positiven X-Achse. Der Zeigefinger der linken Hand weist in Richtung der positiven Y-Achse. Der Mittelfinger der linken Hand wird, von der Handfläche aus gesehen, nach vorne gestreckt. Damit weist er in Richtung der positiven Z-Achse.

Achsen Der positive Teil der Achse (oder kurz: die positive Achse) ist derjenige Teil mit dem farbigen Kegel und der Achsenbezeichnung (x, y oder z). Die negative Achse ist diejenige mit dem nicht farbigen Kegel.

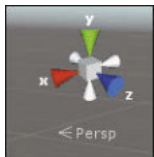


Abbildung 8.5 Gizmo

Isometrisch Klicken Sie einmal auf den Würfel im Zentrum des Gizmos: Die Darstellungsart wechselt zu ISOMETRISCH. Alle 3D-Objekte werden unabhängig von der Entfernung zum Betrachter gleich groß dargestellt. Der räumliche Effekt geht verloren. Ein erneuter Klick wechselt wieder zur perspektivischen Darstellungsart.

Klick auf Achse Klicken Sie nacheinander auf die drei positiven Achsen des Gizmos. Anschließend betrachten wir die Szene aus der jeweiligen positiven Achsenrichtung. Klicken Sie nacheinander auf die drei negativen Achsen. Anschließend betrachten wir die Szene aus der jeweiligen negativen Achsenrichtung.

Ansicht ändern Versuchen Sie als Letztes, durch Verschieben, Drehen, Zoomen und Anklicken der Achsen ungefähr die Ansicht in Abbildung 8.6 herzustellen: Die positive X-Achse weist nach rechts, die positive Y-Achse weist nach oben. Die Ansicht wird ganz leicht gedreht, sodass die linke und die obere Seite

des Würfels zu sehen sind. Das ist beim ersten Mal nicht ganz einfach, stellt aber eine gute Übung dar.

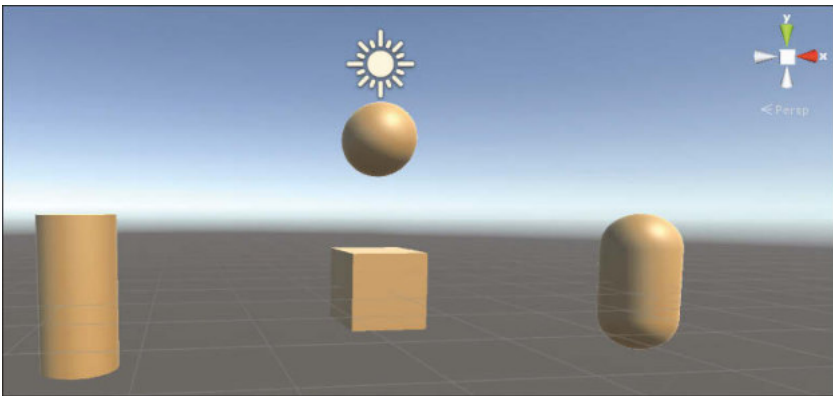


Abbildung 8.6 Gewünschte Ansicht

Bei Bedarf können Sie nun auch die Ansicht in der GAME VIEW gleichartig gestalten. Wählen Sie dazu in der HIERARCHY VIEW das Spielobjekt Main Camera aus, und rufen Sie den Menüpunkt GAMEOBJECT • ALIGN WITH VIEW auf.

Align with View

8.2 Verschieben und Drehen

Bei der Verschiebung und besonders bei der Drehung von 3D-Objekten sollten Sie eine gute Vorstellung des dreidimensionalen Raums haben. Es folgen einige Bewegungen der Objekte mithilfe von Programmcode im Projekt *DreiDimensionen*.

8.2.1 Spielobjekte drehen

Den 3D-Objekten werden *Drehmomente* zugeordnet. Damit werden Drehungen der 3D-Objekte um bestimmte Achsen des dreidimensionalen Koordinatensystems verdeutlicht.

Was ist ein Drehmoment? Nehmen wir an, die vorhandenen 3D-Objekte könnten sich nur drehen, aber ihre Position nicht verändern. Nehmen wir weiterhin an, Sie stecken eine lange Stange durch ein 3D-Objekt, zum Beispiel durch den Würfel, der im Zentrum steht, und zwar genau entlang der

Drehmoment

Y-Achse. Falls Sie in X-Richtung außerhalb des Würfels gegen die Stange drücken, dreht sich der Würfel um seine Z-Achse: Sie üben ein Drehmoment aus. Das Drehmoment ergibt sich aus der Formel *Kraft mal Hebelarm*. Es gilt:

- Je größer Ihre Kraft ist, desto größer ist das Drehmoment.
- Je größer die Entfernung Ihres Angriffspunkts vom Würfel ist, desto größer ist Ihr Hebelarm und damit auch das Drehmoment.

Richtige Kategorie wählen Dem Zylinder, dem Würfel und der Kapsel wird jeweils eine Rigidbody-Komponente zugeordnet. Das geschieht in der INSPECTOR VIEW mithilfe der Schaltfläche ADD COMPONENT. Achten Sie bei 3D-Objekten darauf, die Komponente RIGIDBODY aus der Kategorie PHYSICS zu wählen. Wählen Sie *nicht* die Komponente RIGIDBODY 2D aus der Kategorie PHYSICS 2D, denn sie ist nur für zweidimensionale Objekte sinnvoll.

Keine Schwerkraft Entfernen Sie in der Komponente RIGIDBODY die Markierung bei der Eigenschaft USE GRAVITY, sodass die 3D-Objekte keiner Schwerkraft unterliegen.

Kein Drehwiderstand Setzen Sie den Wert für ANGULAR DRAG auf 0, damit einer Drehung kein Widerstand entgegengesetzt wird.

Klasse »Cube« Es folgt der Code für die Drehung des Würfels um die Y-Achse:

```
using UnityEngine;
public class Cube : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.U))
            GetComponent<Rigidbody>().AddTorque(0, 5, 0);
        else if (Input.GetKeyDown(KeyCode.I))
            GetComponent<Rigidbody>().AddTorque(0, -5, 0);
    }
}
```

Listing 8.2 Klasse »Cube«

AddTorque() Falls der Benutzer die Taste **U** oder die Taste **I** betätigt, wird die Methode AddTorque() der Komponente RIGIDBODY aufgerufen. Damit wird dem 3D-Objekt ein Drehmoment zugeordnet, und zwar um die Achse, die mit den nachfolgenden Parametern für x, y und z festgelegt wird. In der

Klasse Cube handelt es sich jeweils um die Y-Achse. Anschließend dreht sich das 3D-Objekt immer weiter, da der Drehung kein Widerstand entgegengesetzt wird (ANGULAR DRAG = 0).

Ein positiver Wert erzeugt eine Drehung in positiver Drehrichtung, ein negativer Wert entsprechend in negativer Drehrichtung. Zur Verdeutlichung der Drehrichtung um eine Achse kann wiederum die linke Hand genutzt werden: Der Daumen der linken Hand weist in Richtung der positiven Achse, um die gedreht wird. Die restlichen Finger der Hand werden leicht gekrümmt. Die Fingerspitzen weisen in die positive Drehrichtung um die jeweilige Achse.

Drehrichtung

Falls der Benutzer die Tasten **U** oder **I** mehrfach betätigt, werden weitere Drehmomente addiert. Das 3D-Objekt dreht sich dann schneller beziehungsweise langsamer oder ändert seine Drehrichtung.

Drehung ändern

Es gibt zwei weitere, gleichartig aufgebaute Klassen:

- In der Klasse Cylinder führt die Betätigung der Taste **Y** oder der Taste **X** zu einer Drehung um die X-Achse, und zwar mithilfe der Aufrufe AddTorque(5, 0, 0) beziehungsweise AddTorque(-5, 0, 0).
- In der Klasse Capsule führt die Betätigung der Taste **A** oder der Taste **S** zu einer Drehung um die Z-Achse, und zwar mithilfe der Aufrufe AddTorque(0, 0, 5) beziehungsweise AddTorque(0, 0, -5).

Zylinder drehen

Kapsel drehen

Die drei C#-Scripte sind den Spielobjekten Cube, Cylinder und Capsule zugeordnet. Betätigen Sie die genannten Tasten, und führen Sie sich anhand der Drehungen die Achsen, Drehmomente und Drehrichtungen vor Augen. In Abbildung 8.7 sehen Sie eine Momentaufnahme, nachdem alle drei Objekte in Drehung versetzt wurden.

Testen

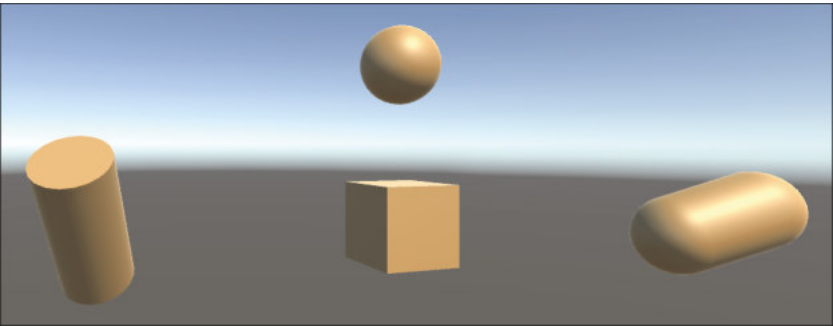


Abbildung 8.7 Zylinder, Würfel und Kapsel drehen sich.

8.2.2 Animiert verschieben

Klasse »Sphere« Ein Spielobjekt soll sich mithilfe von Programmcode wie bei einer Animation in einer bestimmten Zeit von einem Startpunkt A zu einem Zielpunkt B bewegen. Zu diesem Zweck können Sie den folgenden Code nutzen, der die Klasse Sphere ergänzt:

```
using UnityEngine;
public class Sphere : MonoBehaviour
{
    ...
    bool bewegung;
    float zeitGesamt;
    Vector3 startPunkt;
    Vector3 streckeGesamt;
    float bewegungZeitStart;

    void Start()
    {
        bewegung = false;
        zeitGesamt = 5;
        startPunkt = new Vector3(0, 2, 0);
        Vector3 zielPunkt = new Vector3(0, -2, 0);
        streckeGesamt = zielPunkt - startPunkt;
    }

    void Update()
    {
        if ...
        else if (Input.GetKeyDown (KeyCode.H) && !bewegung)
        {
            bewegung = true;
            bewegungZeitStart = Time.time;
        }

        if (bewegung)
        {
            float zeitAnteil =
                (Time.time - bewegungZeitStart) / zeitGesamt;
            Vector3 streckeAnteil = zeitAnteil * streckeGesamt;
            transform.position = startPunkt + streckeAnteil;
            if(zeitAnteil >= 1)
                bewegung = false;
        }
    }
}
```

```
}
}
}
```

Listing 8.3 Klasse »Sphere«, Verschiebung von A nach B

Zunächst werden einige Variablen deklariert, die innerhalb der gesamten Klasse gelten. Sie werden innerhalb der Methoden erläutert.

In der Methode Start() werden die Daten für die gewünschte Bewegung festgelegt. Die boolesche Variable bewegung wird auf false gesetzt. Sie steht nur während der Bewegung auf true. Die Gesamtzeit für die Bewegung wird auf fünf Sekunden gesetzt. Den Variablen für den Startpunkt und den Zielpunkt der Bewegung werden Vector3-Werte zugewiesen. Die Gesamtstrecke ist ebenfalls ein Vector3-Wert. Sie entspricht dem Vektor vom Startpunkt zum Zielpunkt.

In der Methode Update() wird die Bewegung nach Betätigung der Taste **H** gestartet, falls sie zurzeit nicht ausgeführt wird. Dabei wird der Startzeitpunkt festgehalten.

Nach dem Start der Bewegung wird das 3D-Objekt kontinuierlich weiterbewegt. Es wird der Anteil an der Gesamtzeit berechnet, der seit dem Startzeitpunkt vergangen ist. Daraus wird der zugehörige Anteil an der Gesamtstrecke berechnet. Anschließend wird das 3D-Objekt auf die Position gesetzt, die sich aus dem Startpunkt und diesem Streckenanteil ergibt. Die Bewegung wird gestoppt, falls die Gesamtzeit abgelaufen ist. Das 3D-Objekt hat in diesem Fall den Zielpunkt erreicht.

Deaktivieren Sie im UNITY EDITOR beim Spielobjekt Sphere die Komponente SPHERE COLLIDER, indem Sie die Markierung oben links in der Komponente entfernen. Starten Sie die Bewegung. Da das Spielobjekt Sphere keinen aktiven Collider besitzt, bewegt es sich durch das Spielobjekt Cube hindurch, auch wenn es sich zurzeit drehen sollte (siehe Abbildung 8.8). Stellen Sie unterschiedliche Zielpunkte und Laufzeiten ein, und starten Sie die Bewegung erneut.

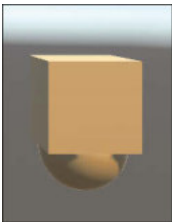


Abbildung 8.8 Würfel bewegt sich von A nach B.

Gesamtzeit und Gesamtstrecke

Anteilige Verschiebung

Testen

8.2.3 Kamera bewegen

Sie können nicht nur die Spielobjekte, sondern auch die Kamera bewegen. Damit vergrößern Sie das mögliche Spielfeld. Zudem wird das Spiel anschaulicher. Folgende Bewegungen werden verdeutlicht:

- Die Kamera wird geradlinig bewegt.
- Die Kamera wird um sich selber gedreht.
- Die Kamera wird um einen bestimmten Punkt gedreht.

Zoomen In Kapitel 14 finden Sie außerdem ein Projekt, bei dem der Benutzer selber mit der Kamera herauszoomen und wieder hineinzoomen kann.

Klasse
»KameraBewegen« Es folgt der Code der Klasse KameraBewegen:

```
using UnityEngine;
public class KameraBewegen : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown (KeyCode.C))
            transform.Translate (0.2f, -0.1f, 0.7f);
        else if (Input.GetKeyDown (KeyCode.V))
            transform.Translate (-0.2f, 0.1f, -0.7f);
        else if (Input.GetKeyDown (KeyCode.O))
            transform.Rotate(0, 5, 0);
        else if (Input.GetKeyDown (KeyCode.P))
            transform.Rotate(0, -5, 0);
        else if (Input.GetKeyDown (KeyCode.J))
            transform.RotateAround (new Vector3 (-4, 0, 0),
                                    new Vector3 (0, 1, 0), 5);
        else if (Input.GetKeyDown (KeyCode.K))
            transform.RotateAround (new Vector3 (-4, 0, 0),
                                    new Vector3 (0, 1, 0), -5);
    }
}
```

Listing 8.4 Klasse »KameraBewegen«

Name der Klasse Die Klasse wird ganz bewusst nicht Camera genannt, weil es bereits eine gleichnamige interne Klasse des Spielobjekts Main Camera gibt.

Translate() In der Klasse KameraBewegen führt die Betätigung der Taste [C] oder der Taste [V] zu einer kurzen geradlinigen Bewegung der Kamera auf der Linie

zwischen der Startposition der Kamera und dem Nullpunkt des Koordinatensystems. Die Kamera ist zu Beginn bei -2/1/-7 positioniert. Die Parameter der Methode Translate() stehen in demselben Verhältnis zueinander.

Die Betätigung der Taste [O] oder der Taste [P] führt zu einer 5-Grad-Drehung der Kamera um ihre eigene Y-Achse herum. Die Parameter der Methode Rotate() geben den Winkel an, um den das Spielobjekt um die eigene X-, Y- und Z-Achse weitergedreht wird.

Rotate()

Die Betätigung der Taste [J] oder der Taste [K] führt zu einer 5-Grad-Drehung der Kamera um eine Achse herum, die parallel zur Y-Achse des Koordinatensystems durch die Mitte des Zylinders verläuft. Die drei Parameter der Methode RotateAround() geben Folgendes an:

RotateAround()

- den Punkt, um den gedreht wird (hier die Position des Zylinders)
- die Lage der Drehachse, die durch den genannten Punkt verläuft (hier eine Achse parallel zur Y-Achse)
- den Winkel, um den das Spielobjekt um die genannte Achse weitergedreht wird

Das C#-Script ist dem Spielobjekt Main Camera zugeordnet. Betätigen Sie die genannten Tasten, und führen Sie sich die Bewegung der Kamera und die Auswirkungen vor Augen. In Abbildung 8.9 sehen Sie die Ansicht, nachdem die Taste [J] mehrmals betätigt wurde.

Testen

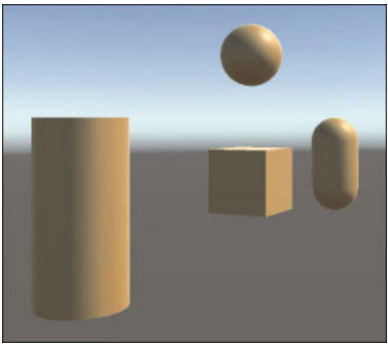


Abbildung 8.9 Kamera wird um Zylinder gedreht.

8.2.4 Animiert drehen

Ein Spielobjekt soll sich mithilfe von Programmcode wie bei einer Animation in einer bestimmten Zeit und von einem Startwinkel A ausgehend so lange um eine bestimmten Achse drehen, bis ein Zielwinkel B erreicht ist.

Klasse
»KameraBewegen«

Das wird mithilfe des nachfolgenden Codes für das Spielobjekt Main Camera realisiert:

```
using UnityEngine;
public class KameraBewegen : MonoBehaviour
{
    bool bewegung;
    float zeitGesamt;
    float winkelGesamt;
    float bewegungZeitStart;
    float zeitAnteilAlt;

    void Start()
    {
        bewegung = false;
        zeitGesamt = 3;
        winkelGesamt = 90;
        zeitAnteilAlt = 0;
    }

    void Update()
    {
        if...
        else if (Input.GetKeyDown(KeyCode.L) && !bewegung)
        {
            bewegung = true;
            bewegungZeitStart = Time.time;
        }

        if (bewegung)
        {
            float zeitAnteil =
                (Time.time - bewegungZeitStart) / zeitGesamt;
            float winkelAenderung =
                (zeitAnteil - zeitAnteilAlt) * winkelGesamt;
            transform.RotateAround(Vector3.zero,
                new Vector3(0, 1, 0), winkelAenderung);
            zeitAnteilAlt = zeitAnteil;
            Debug.Log(transform.eulerAngles.y);
            if(zeitAnteil >= 1)
                bewegung = false;
        }
    }
}
```

```
}
}
}
```

Listing 8.5 Klasse »KameraBewegen«, mit Drehung von A nach B

Der Ablauf ähnelt demjenigen bei der Verschiebung von einem Startpunkt A zu einem Zielpunkt B aus Abschnitt 8.2.2. Zunächst werden einige Variablen deklariert, die innerhalb der gesamten Klasse gelten. Sie werden innerhalb der Methoden erläutert.

In der Methode `Start()` werden die Daten für die gewünschte Bewegung festgelegt. Die boolesche Variable `bewegung` wird auf `false` gesetzt. Sie steht nur während der Bewegung auf `true`. Die Gesamtzeit für die Bewegung wird auf drei Sekunden gesetzt und der Gesamtwinkel auf 90 Grad. Da sich der Winkel bei der Methode `RotateAround()` immer relativ zum vorhergehenden Winkel ändert, muss der Zeitpunkt der vorhergehenden Änderung gespeichert werden (hier in der Variablen `zeitAnteilAlt`).

Gesamtzeit und Gesamtwinkel

In der Methode `Update()` wird die Bewegung nach Betätigung der Taste `L` gestartet, falls sie zurzeit nicht ausgeführt wird. Dabei wird der Startzeitpunkt festgehalten.

Nach dem Start der Bewegung wird das 3D-Objekt kontinuierlich weitergedreht. Es wird der Anteil an der Gesamtzeit berechnet, der seit dem Startzeitpunkt vergangen ist. Anschließend wird der Winkel berechnet, um den sich das Spielobjekt seit der letzten Änderung weiterdrehen soll. Diese Drehung wird mithilfe der Methode `RotateAround()` ausgeführt und findet hier um die Y-Achse herum statt, die durch das Zentrum des Koordinatensystems verläuft. Der aktuelle Zeitanteil wird zur Durchführung der nächsten Änderung gespeichert. Die Bewegung wird gestoppt, falls die Gesamtzeit abgelaufen ist.

Anteilige Drehung

Der aktuelle Drehwinkel um die Y-Achse wird mithilfe der statischen Methode `Log()` der Klasse `Debug` zur Kontrolle ausgegeben. Die Transform-Komponente besitzt die Untereigenschaft `eulerAngles`. Diese hat den Typ `Vector3` und gibt die Drehung um die X-Achse, um die Y-Achse und um die Z-Achse in einem Winkel von 0 bis 360 Grad an.

Drehwinkel

Starten Sie die Bewegung. Stellen Sie unterschiedliche Zielwinkel und Laufzeiten ein, und starten Sie die Bewegung erneut. Beachten Sie auch die Kontrollausgabe des Drehwinkels.

Testen

8.2.5 Übersicht

Alle Tastencodes Es folgt eine Tabelle mit allen Tastencodes im Projekt *DreiDimensionen* in der Reihenfolge ihres Einsatzes in diesem Kapitel:

Taste	Erläuterung
B	Kugel erhält braune Oberfläche.
G	Kugel erhält grüne Oberfläche.
U	Würfel erhält positives Drehmoment um die Y-Achse.
I	Würfel erhält negatives Drehmoment um die Y-Achse.
Y	Zylinder erhält positives Drehmoment um die X-Achse.
X	Zylinder erhält negatives Drehmoment um die X-Achse.
A	Kapsel erhält positives Drehmoment um die Z-Achse.
S	Kapsel erhält negatives Drehmoment um die Z-Achse.
H	Kugel wird animiert von Punkt A nach Punkt B verschoben.
C	Kamera wird verschoben, vom Betrachter weg.
V	Kamera wird verschoben, zum Betrachter hin.
O	Kamera wird positiv um ihre Y-Achse gedreht.
P	Kamera wird negativ um ihre Y-Achse gedreht.
J	Kamera wird positiv um eine Achse gedreht, die parallel zur Y-Achse steht.
K	Kamera wird negativ um eine Achse gedreht, die parallel zur Y-Achse steht.
L	Kamera wird animiert um einen Punkt von Winkel A nach Winkel B gedreht.

Tabelle 8.1 Tastencodes im Projekt

Time.deltaTime Möglicherweise vermissen Sie bei den Bewegungen und Drehungen den Faktor `Time.deltaTime`. Im vorliegenden Projekt gibt es aber nur

- einmalige Bewegungen auf eine neue Position,
- einmalige Drehungen auf eine neue Rotationsposition und
- kontinuierliche Änderungen mithilfe einer eigenen Zeitsteuerung.

Daher wird der Wert von `Time.deltaTime` nicht benötigt.

Auf einen Blick

1	Das erste 2D-Projekt	25
2	Spielen Sie ein 2D-Jump&Run-Spiel	39
3	Entwickeln Sie ein 2D-Jump&Run-Spiel	49
4	Ein 2D-Breakout-Spiel	95
5	Ein 2D-Spiel für zwei Spieler	129
6	Ein Gedächtnistrainer als 2D-Projekt	149
7	Ein 2D-Space-Shooter	167
8	Das erste 3D-Projekt	187
9	Eine 3D-Animation	203
10	Ein 3D-Balancer	217
11	Ein 3D-Tetris	229
12	Ein Kopfrechentrainer als 3D-Projekt	241
13	Golf spielen auf einem 3D-Terrain	253
14	Jagen auf einem 3D-Terrain	279
15	Eine Schlange aus 3D-Joints	301
16	Ein Renntraining und ein Autorennen	315
17	Erkunden Sie das Verlies	346
18	Ein Programmierkurs in C#	378
19	Speichern Sie eine Highscore-Liste	417
20	Arbeiten Sie mit mehreren Szenen	427
21	Weitere Plattformen	434

Inhalt

Materialien zum Buch	16
Einführung	17
1 Das erste 2D-Projekt	25
1.1 Erstellung eines neuen Projekts	25
1.2 Wichtige Bereiche im Unity Editor	26
1.3 Das Spielobjekt »Main Camera«	28
1.4 Assets importieren	28
1.5 Spielobjekte einfügen	29
1.6 Ändern der Hierarchie	30
1.7 Eine Szene speichern	32
1.8 Die Komponente »Transform«	32
1.8.1 Die Eigenschaften der Transform-Komponente	33
1.8.2 Werte in der »Inspector View« ändern	34
1.9 Die Ansicht in der »Scene View«	35
1.9.1 Positionswerte mithilfe der Maus ändern	36
1.9.2 Rotationswerte mithilfe der Maus ändern	37
1.9.3 Scale-Werte mithilfe der Maus ändern	38
2 Spielen Sie ein 2D-Jump&Run-Spiel	39
2.1 Starten Sie das Spiel	39
2.2 Wie geht das Spiel?	40
2.3 Unsere ersten Unity-Elemente	41
2.3.1 Assets	42
2.3.2 Spielobjekte	44

3	Entwickeln Sie ein 2D-Jump&Run-Spiel	49
3.1	Erzeugen Sie Projekt und Assets	49
3.2	Fügen Sie Spielobjekte ein	50
3.2.1	Erzeugen Sie das Spielfeld	50
3.2.2	Setzen Sie den Spieler auf den Boden	52
3.3	Erstellen Sie den Spielablauf	55
3.3.1	Führen Sie die Klasse »Spieler« ein	55
3.3.2	Bewegen Sie den Spieler	57
3.3.3	Begrenzen Sie die Bewegung	62
3.3.4	Treffen Sie den Gewinn	64
3.3.5	Vermeiden Sie die Gefahren	67
3.3.6	Die geschweiften Klammern	69
3.3.7	Die Gefahren bewegen sich	70
3.4	Gestalten Sie die Benutzeroberfläche	72
3.4.1	Erstellen Sie die erste Anzeige	73
3.4.2	Sammeln Sie Punkte	74
3.4.3	Verlieren Sie Leben	77
3.4.4	Messen Sie die Spielzeit	78
3.4.5	Speichern Sie Werte dauerhaft	81
3.4.6	Geben Sie den Benutzern Hinweise	83
3.4.7	Starten Sie ein neues Spiel	87
3.4.8	Beenden Sie die Anwendung	90
3.4.9	Ideen für Ihre Erweiterungen	91
3.5	Erzeugen Sie eine ausführbare Version	92
3.6	Projekte umbenennen oder kopieren	94
4	Ein 2D-Breakout-Spiel	95
4.1	Führen Sie das Spiel aus	95
4.2	Erzeugen Sie Projekt und Assets	96
4.2.1	Fügen Sie ein Audio-Asset ein	97
4.2.2	Erstellen Sie ein 2D-Material	97
4.2.3	Lernen Sie 2D-Materialien kennen	98
4.2.4	Erzeugen Sie ein Prefab	100

4.3	Fügen Sie Spielobjekte ein	101
4.3.1	Füllen Sie das Spielfeld	101
4.3.2	Erzeugen Sie einen Ziegel	102
4.3.3	Wiederholen Sie den Vorgang	103
4.3.4	Wiederholen Sie die Wiederholung	105
4.4	Erstellen Sie den Spielablauf	106
4.4.1	Senden Sie den Ball ab	106
4.4.2	Bewegen Sie den Spieler	109
4.4.3	Sammeln Sie Punkte	110
4.4.4	Verlieren Sie Leben	113
4.5	Gestalten Sie die Benutzeroberfläche	116
4.5.1	Exportieren Sie ein Asset Package	116
4.5.2	Importieren Sie ein Asset Package	117
4.5.3	Passen Sie die Benutzeroberfläche an	118
4.5.4	Punkte, Leben und Infos anzeigen	119
4.5.5	Messen Sie die Spielzeit	121
4.5.6	Zeigen Sie die vorherige Zeit an	122
4.5.7	Starten Sie ein neues Spiel	124
4.5.8	Beenden Sie die Anwendung	127
4.5.9	Ideen für Ihre Erweiterungen	128
5	Ein 2D-Spiel für zwei Spieler	129
5.1	Führen Sie das Spiel aus	129
5.2	Bereiten Sie das Spiel vor	131
5.2.1	Erzeugen Sie Projekt und Assets	131
5.2.2	Erzeugen Sie Spielfeld und UI	132
5.2.3	Positionieren Sie die Hindernisse	134
5.3	Erstellen Sie den Spielablauf	136
5.3.1	Führen Sie den Aufschlag aus	137
5.3.2	Bewegen Sie die Spieler vertikal	139
5.3.3	Bewegen Sie die Spieler horizontal	141
5.3.4	Sammeln Sie Punkte	143
5.3.5	Eine kleine Übung	146
5.3.6	Ideen für Ihre Erweiterungen	146
5.4	Künstliche Intelligenz	146

6	Ein Gedächtnistrainer als 2D-Projekt	149
6.1	Führen Sie das Training aus	149
6.2	Bereiten Sie das Training vor	150
6.2.1	Erzeugen Sie die Benutzeroberfläche	150
6.3	Das Training für drei Zahlen	151
6.3.1	Verteilen Sie die Zahlen	151
6.3.2	Vermeiden Sie doppelte Positionen	154
6.3.3	Löschen Sie die Zahlen	156
6.3.4	Prüfen Sie die Reihenfolge	157
6.4	Die Erweiterung des Trainings	159
6.4.1	Machen Sie das Training leichter	160
6.4.2	Machen Sie das Training schwerer	161
6.4.3	Optimieren Sie das Training	164
6.4.4	Ideen für Ihre Erweiterungen	166
7	Ein 2D-Space-Shooter	167
7.1	Bereiten Sie das Spiel vor	168
7.1.1	Gestalten Sie die beiden Explosions-Prefabs	168
7.1.2	Erzeugen Sie Ihr Raumschiff und die Geschosse	170
7.1.3	Erstellen Sie die anderen Raumschiffe	171
7.1.4	Gestalten Sie die Energieanzeige mit Layern	172
7.1.5	Erstellen Sie die Benutzeroberfläche	173
7.2	Erstellen Sie den Spielablauf	173
7.2.1	Bewegen Sie Ihr Raumschiff, und feuern Sie	173
7.2.2	Bewegen Sie die Geschosse nach dem Abfeuern	176
7.2.3	Bewegen Sie die anderen Raumschiffe	176
7.2.4	Lassen Sie die Raumschiffe explodieren	178
7.2.5	Kollidieren Sie mit den anderen Raumschiffen	180
7.2.6	Führen Sie weitere Änderungen der Energie herbei	182
7.2.7	Messen Sie die Zeit, und beenden Sie das Spiel	184
7.2.8	Eine kleine Übung	186
7.2.9	Ideen für Ihre Erweiterungen	186

8	Das erste 3D-Projekt	187
8.1	Grundlagen eines 3D-Projekts	187
8.1.1	Kamera, Skybox und Licht	187
8.1.2	Einfache 3D-Objekte	188
8.1.3	Oberflächenmaterial erstellen und zuordnen	189
8.1.4	Oberflächenmaterial wechseln	190
8.1.5	Ansicht in der »Scene View« gestalten	191
8.2	Verschieben und Drehen	193
8.2.1	Spielobjekte drehen	193
8.2.2	Animiert verschieben	196
8.2.3	Kamera bewegen	198
8.2.4	Animiert drehen	199
8.2.5	Übersicht	202
9	Eine 3D-Animation	203
9.1	Schaffen Sie die Voraussetzungen	203
9.1.1	Betrachten Sie die fertige Animation	203
9.1.2	Bauen Sie das Beispiel auf	204
9.2	Erstellen Sie die Animation	205
9.2.1	Legen Sie die Animation an	205
9.2.2	Drehen Sie das rechte Bein	207
9.2.3	Erstellen Sie weitere Keyframes	208
9.2.4	Stellen Sie die Keyframes ein	208
9.2.5	Verschieben Sie das rechte Bein	209
9.3	Arbeiten Sie mit dem »Animator Controller«	210
9.3.1	Gestalten Sie die States	210
9.3.2	Erstellen Sie die Parameter	212
9.3.3	Erzeugen Sie die Transitions	212
9.4	Fügen Sie das C#-Script hinzu	213
9.4.1	Verbinden Sie Bewegung und Animation	213
9.4.2	Vervollständigen Sie die Animation	215
9.4.3	Ideen für Ihre Erweiterungen	216

10

Ein 3D-Balancer

217

10.1

Führen Sie das Spiel aus

217

10.2

Bereiten Sie das Spiel vor

218

10.2.1

Erzeugen Sie Projekt und Assets

218

10.2.2

Erzeugen Sie Spielfeld und UI

219

10.2.3

Relative Transform-Werte

220

10.3

Erstellen Sie den Spielablauf

222

10.3.1

Drehen Sie die Platte

222

10.3.2

Bewegen Sie die Kugel und die Kamera

224

10.3.3

Ändern Sie die Punktzahl

226

10.3.4

Ideen für Ihre Erweiterungen

228

11

Ein 3D-Tetris

229

11.1

Führen Sie das Spiel aus

229

11.2

Bereiten Sie das Spiel vor

230

11.2.1

Erzeugen Sie Projekt und Assets

230

11.2.2

Erzeugen Sie Spielfeld und UI

231

11.2.3

Erstellen Sie das Würfel-Prefab

232

11.3

Erstellen Sie den Spielablauf

232

11.3.1

Bewegen Sie die Würfel

232

11.3.2

Eine »generische Liste«

234

11.3.3

Fügen Sie Elemente zur Liste hinzu

235

11.3.4

Entfernen Sie Elemente aus der Liste

237

11.3.5

Eine kleine Übung

240

11.3.6

Ideen für Ihre Erweiterungen

240

12

Ein Kopfrechentainer als 3D-Projekt

241

12.1

Führen Sie das Training aus

241

12.2

Bereiten Sie das Training vor

242

12.3

Erstellen Sie den Trainingsablauf

243

12.3.1

Erzeugen Sie die Aufgabe und die Lösungen

243

12.3.2

Mischen Sie die Lösungen

246

12.3.3

Sammeln Sie Punkte

248

12.3.4

Verlieren Sie Leben

250

12.3.5

Ideen für Ihre Erweiterungen

252

13

Golf spielen auf einem 3D-Terrain

253

13.1

Führen Sie das Spiel aus

253

13.2

Bereiten Sie das Spiel vor

254

13.2.1

Erzeugen Sie Projekt und Landschaft

255

13.2.2

Weisen Sie der Landschaft eine Textur zu

255

13.2.3

Erstellen Sie die drei Ebenen

257

13.2.4

Fügen Sie den Rand hinzu

260

13.2.5

Erstellen Sie die beiden Rampen

261

13.2.6

Setzen Sie Spieler und Ziel in die Landschaft

263

13.2.7

Arbeiten Sie mit einem »Physic Material«

264

13.3

Erstellen Sie den Spielablauf

265

13.3.1

Schlagen Sie den Spielball

265

13.3.2

Versetzen Sie das Ziel

268

13.3.3

Vermeiden Sie den Verlust des Spielballs

269

13.3.4

Ideen für Ihre Erweiterungen

271

13.4

Ein weiteres Terrain

271

13.4.1

Erzeugen Sie zehn Ebenen

272

13.4.2

Fügen Sie den linken und den rechten Rand hinzu

273

13.4.3

Fügen Sie den unteren und den oberen Rand hinzu

274

13.4.4

Erzeugen Sie die erste Rampe

275

13.4.5

Erstellen Sie alle Rampen links

275

13.4.6

Erstellen Sie alle Rampen rechts

276

13.4.7

Setzen Sie die Positionen

277

14

Jagen auf einem 3D-Terrain

279

14.1

Führen Sie das Spiel aus

279

14.2

Bereiten Sie das Spiel vor

282

14.2.1	Erzeugen Sie Projekt und Landschaft	282
14.2.2	Steuern Sie den Zufall	282
14.2.3	Erzeugen Sie die weiteren Spielobjekte	285
14.2.4	Erstellen Sie die drei Prefabs	287
14.2.5	Zoomen Sie mithilfe eines Sliders	288
14.3	Erstellen Sie den Spielablauf	290
14.3.1	Bewegen Sie den Jäger	290
14.3.2	Treffen Sie die Ziele	292
14.3.3	Die Ziele starten eine Abwehr	295
14.3.4	Die Abwehr wird gefährlich	296
14.3.5	Messen Sie die Zeit	299
14.3.6	Ideen für Ihre Erweiterungen	300
15	Eine Schlange aus 3D-Joints	301
15.1	Führen Sie das Spiel aus	301
15.2	Bereiten Sie das Spiel vor	302
15.2.1	Erzeugen Sie die Assets und die Platte	302
15.2.2	Erstellen Sie die Schlange und ihre Beute	303
15.2.3	Stellen Sie die gelenkigen Verbindungen her	304
15.3	Erstellen Sie den Spielablauf	305
15.3.1	Bewegen Sie die Schlange	306
15.3.2	Treffen Sie die Beute	307
15.3.3	Verkürzen Sie die Schlange	308
15.3.4	Zählen Sie die Punkte	310
15.3.5	Die Segmente treffen den Rand	311
15.3.6	Messen Sie die Zeit	312
15.3.7	Ideen für Ihre Erweiterungen	314
16	Ein Renntraining und ein Autorennen	315
16.1	Führen Sie das Renntraining aus	315
16.2	Führen Sie das Autorennen aus	317
16.3	Bereiten Sie das Renntraining vor	319

16.3.1	Erzeugen Sie das Projekt und die Fahrbahn	319
16.3.2	Konstruieren Sie das Fahrzeug	320
16.3.3	Fügen Sie die Wheel Collider hinzu	321
16.4	Erstellen Sie den Ablauf des Renntrainings	323
16.4.1	Beschleunigen Sie das Fahrzeug	323
16.4.2	Lenken Sie das Fahrzeug	324
16.4.3	Folgen Sie dem Fahrzeug mit der Kamera	326
16.4.4	Bauen Sie die Begrenzungen auf	328
16.4.5	Eine »Lichtschranke« an der Startlinie	330
16.4.6	Messen Sie die Rundenzeiten	332
16.5	Erweitern Sie das Renntraining zum Autorennen	334
16.5.1	Erzeugen Sie das zweite Fahrzeug	335
16.5.2	Steuern Sie die Fahrzeuge getrennt	336
16.5.3	Teilen Sie den Bildschirm auf	338
16.5.4	Eine dritte Kamera für den Überblick	339
16.5.5	Getrennte Rundenzeiten nach einem Countdown	341
16.5.6	Ideen für Ihre Erweiterungen	345
17	Erkunden Sie das Verlies	346
17.1	Führen Sie das Spiel aus	346
17.1.1	Der Ablauf des Spiels	347
17.2	Bereiten Sie das Spiel vor	350
17.2.1	Die Planung des Verlieses	350
17.2.2	Der Aufbau einer Kammer	351
17.2.3	Erstellen Sie die ersten Spielobjekte	353
17.2.4	Bauen Sie das Prefab für die Kammer	353
17.2.5	Die Schlüssel, Kisten und Sperren	356
17.2.6	Gestalten Sie die Benutzeroberfläche	358
17.3	Erstellen Sie den Spielablauf	359
17.3.1	Folgen Sie dem Spieler mit der Kamera	359
17.3.2	Erstellen Sie alle Kammern	361
17.3.3	Konfigurieren Sie die Kammern	363
17.3.4	Gehen Sie durch ein Tor	365
17.3.5	Nehmen Sie den Schlüssel aus einer Schatzkiste	368

17.3.6	Schließen Sie eine Sperre auf	372
17.3.7	Speichern Sie den Spielstand	374
17.3.8	Laden Sie den alten Spielstand	375
17.3.9	Ideen für Ihre Erweiterungen	377
18	Ein Programmierkurs in C#	378
18.1	Das Unity-Projekt »Programmierkurs«	378
18.2	Grundlagen	380
18.2.1	Variablen und Datentypen	380
18.2.2	Rechenoperatoren	383
18.2.3	Division von ganzen Zahlen	385
18.2.4	Verzweigungen	385
18.2.5	Logische Verknüpfungen	388
18.2.6	Schleifen und Zufallszahlen	390
18.3	Datenfelder	394
18.4	Zeichenketten	397
18.5	Methoden	399
18.5.1	Einfache Methode	400
18.5.2	Methode mit Parametern	400
18.5.3	Methode mit Rückgabewert	401
18.5.4	Methode mit Verweis-Parameter	403
18.6	Generische Listen	404
18.6.1	Hilfsmethode »AusgabeListe()«	406
18.6.2	foreach-Schleife	407
18.7	Daten auf der Festplatte	408
18.7.1	Daten speichern	408
18.7.2	Daten laden	409
18.7.3	Kontrolle der Daten	410
18.8	Objektorientierung	411
18.8.1	Die Spielobjekte im »Unity Editor«	412
18.8.2	Die Klasse »Spieler«	413
18.8.3	Änderungen aller Objekte der Klasse	414
18.8.4	Änderungen einzelner Objekte	415

19	Speichern Sie eine Highscore-Liste	417
19.1	Definition der eigenen Klasse	417
19.2	Nutzung der eigenen Klasse	419
19.2.1	Generische Liste erzeugen und füllen	419
19.2.2	Generische Liste anzeigen	421
19.2.3	Einen neuen Eintrag hinzufügen	422
19.2.4	Alles speichern, alles löschen	424
19.2.5	Der Anzeige-Schalter	425
20	Arbeiten Sie mit mehreren Szenen	427
20.1	Der Ablauf des Projekts	427
20.2	Der Aufbau der ersten Szene	428
20.2.1	Die Klasse »Spieler«	428
20.3	Weitere Szenen	432
21	Weitere Plattformen	434
21.1	Installieren Sie Unity unter macOS High Sierra	434
21.2	Erstellen Sie eine Browser-Anwendung	436
21.3	Erstellen Sie eine Android-App	437
21.3.1	Änderungen im Code	438
21.3.2	Stellen Sie die Player Settings ein	439
21.3.3	Weitere Komponenten und Anwendungen	441
21.3.4	Führen Sie den Android-Build durch	441
21.3.5	Starten Sie die App unter Android	441
21.4	Bonusprojekte	442
21.4.1	Bonusprojekt »TomsFrogger«	442
21.4.2	Bonusprojekt »TomsPacman«	443
	Index	445