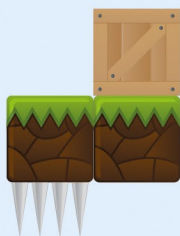
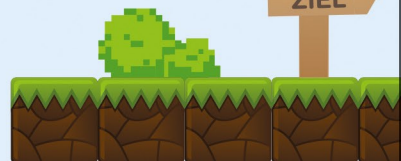
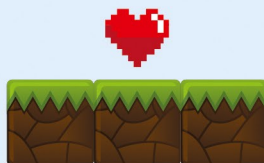


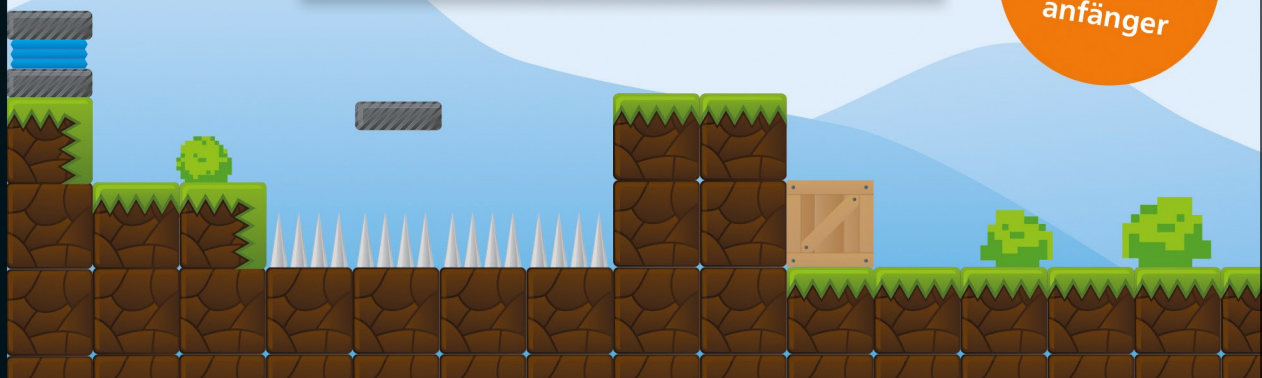
Thomas Theis



Einstieg in Unity

2D- und 3D-Spiele entwickeln

*Ideal für
Programmier-
anfänger*



- ▶ Schritt für Schritt zum eigenen Computerspiel
- ▶ Mit C#-Programmierkurs für Unity-Skripte
- ▶ 18 fertige Spiele als Vorlage für eigene Ideen



Alle Beispielprojekte zum Download



Rheinwerk
Computing

Kapitel 1

Einführung

Unity ist eine Entwicklungsumgebung zur Erstellung und Gestaltung von Computerspielen, sowohl unter Windows als auch unter Linux und unter macOS. Unity-Projekte können darüber hinaus auch zu Lern- und Trainingszwecken genutzt werden.

Es gibt unterschiedliche Lizenzmodelle für Unity. Falls Sie Unity zu Übungs- und Lernzwecken nutzen, können Sie die Lizenz *Unity Personal* kostenfrei verwenden.

1.1 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 Sie, wie die Elemente aufeinander reagieren, besonders unter physikalischen Bedingungen.

Physik

Sie 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 selbstständig eigene Spiele zu entwickeln.

Abläufe gestalten

Wir werden nicht einfach vorgefertigte komplexe Elemente miteinander kombinieren, wie sie z. B. 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

Verständnis

können zudem nur mit externen Programmen erstellt werden und müssen danach zunächst in Unity importiert werden.

2D-Spiele 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 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 Spieleentwicklung konzentrieren und müssen sich nicht gleichzeitig mit der Orientierung, der Drehung und der Perspektive im dreidimensionalen Raum befassen.

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

1.2 Wie entsteht der programmierte Spielablauf?

C# Zur Programmierung der logischen Abläufe in den Unity-Projekten können Sie mit der Programmiersprache C# (sprich: C-Sharp) arbeiten. Alle benötigten Elemente der Sprache lernen Sie im Verlauf der Projekte an der passenden Stelle kennen. Bei Unity wird der Begriff *C#-Scripte* statt des Begriffs *C#-Programme* verwendet. Daher verwende ich diesen Begriff ebenfalls.

Zur Entwicklung der C#-Scripte wird bei der Installation der aktuellen Unity-Versionen eine kostenfreie Version der Entwicklungsumgebung *Visual Studio* der Firma Microsoft angeboten. Alternativ können Sie auch die Entwicklungsumgebung *MonoDevelop* installieren, die aus einer älteren Unity-Version stammt und von den Unity-Entwicklern weiterhin gepflegt und zur Verfügung gestellt wird.

Programmierkurs In Kapitel 19, »Ein Programmierkurs in C#«, finden Sie zusätzlich einen reinen 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. Neben dem Erlernen von C# anhand der Unity-Projekte können Sie den C#-Programmierkurs als Ergänzung für das bessere Verständnis und als Nachschlagewerk nutzen, falls Sie etwas über bestimmte Elemente von C# erfahren möchten.

1.3 Dateieindungen anzeigen lassen

Für eine Installation unter Windows und für die nachfolgende Projektentwicklung ist es 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.

1.4 Unity Hub installieren

Zur Installation und Verwaltung von Unity und zur Verwaltung Ihrer eigenen Unity-Projekte empfehle ich zunächst, die Anwendung *Unity Hub* zu installieren. Nachfolgend nenne ich diese Anwendung vereinfacht den Unity Hub.

Sie finden sie mithilfe Ihres Browsers über die folgende Adresse:

Adresse

<https://unity3d.com/de/get-unity/download>

Betätigen Sie die Schaltfläche UNITY HUB HERUNTERLADEN, siehe Abbildung 1.1.

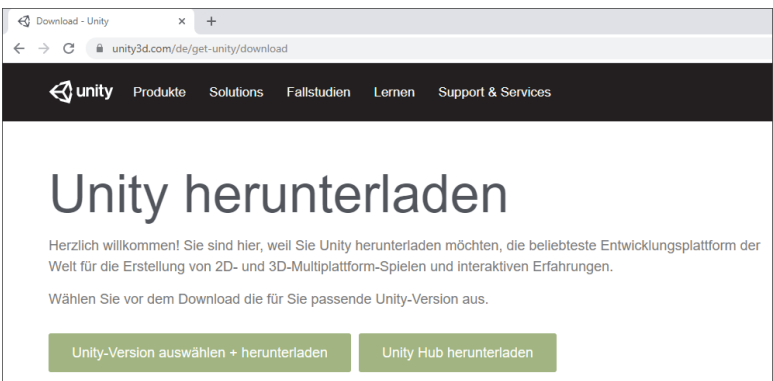


Abbildung 1.1 Unity-Website, Unity Hub herunterladen

Installationsdatei Unter Windows wird die Datei *UnityHubSetup.exe* heruntergeladen, siehe Abbildung 1.2, die Sie anschließend zum Start der Installation aufrufen können. Unter Linux und unter macOS handelt es sich um andere Dateien. Deren Aufruf wird in Abschnitt 22.3, »Unity unter anderen Betriebssystemen«, beschrieben.

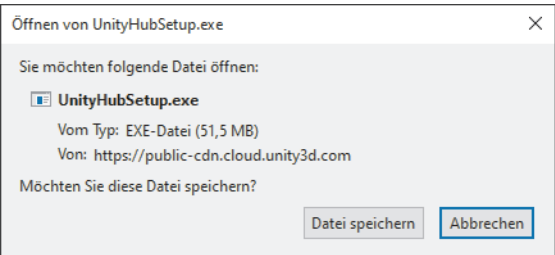


Abbildung 1.2 Installationsprogramm für den Unity Hub

Zu Beginn der Installation müssen die allgemeinen Lizenzbedingungen akzeptiert werden. Im nächsten Dialogfeld wählen Sie das Zielverzeichnis für die Installation aus. Nach der Installation können Sie den Unity Hub ausführen lassen. Zunächst erscheint gegebenenfalls ein Windows-Sicherheitshinweis, in dem Sie den blockierten Zugriff des Unity Hubs explizit zulassen sollten.

Unity-Lizenz Falls Sie Unity erstmals nutzen, haben Sie noch keine Unity-Lizenz. In diesem Fall erscheinen die Einstellungen (engl. *Preferences*) des Dialogfelds UNITY HUB mit der Seite LICENSE MANAGEMENT zur Verwaltung der Unity-Lizenz, siehe Abbildung 1.5.

Zur Erlangung einer Lizenz benötigen Sie als Erstes eine Unity ID. In der Fußzeile erscheint die Meldung YOU NEED TO BE LOGGED IN TO MANAGE YOUR LICENSE. Betätigen Sie daneben den Link LOGIN.

Unity-ID Es erscheint das Dialogfeld UNITY HUB SIGN IN, siehe Abbildung 1.3. Falls Sie noch keine Unity ID haben, betätigen Sie den Link CREATE ONE, um Ihre persönliche Unity ID zu erzeugen. Melden Sie sich anschließend mit Ihrer E-Mail und Ihrem Passwort an.

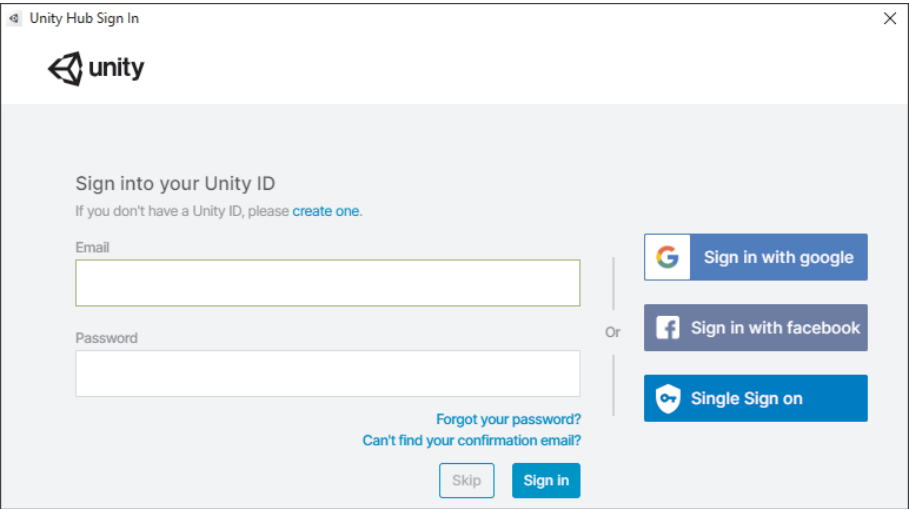


Abbildung 1.3 Anmeldung mit Unity ID

Nach der erfolgreichen Anmeldung betätigen Sie im Dialogfeld PREFERENCES die Schaltfläche ACTIVATE NEW LICENSE. Wählen Sie im Dialogfeld NEW LICENSE ACTIVATION die kostenfreie Lizenz UNITY PERSONAL und danach diejenige Option, die bei Ihnen zutrifft, siehe Abbildung 1.4.

Unity Personal

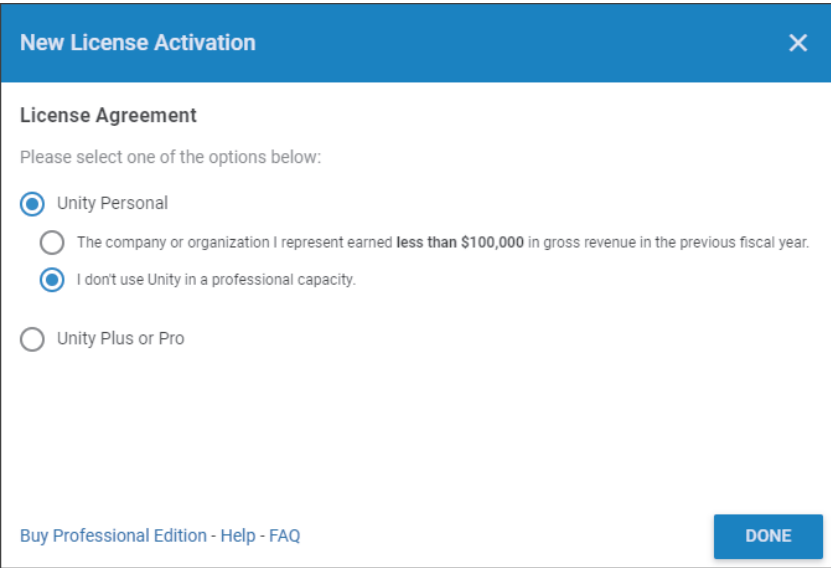


Abbildung 1.4 Auswahl der Lizenz

Nach der Betätigung der Schaltfläche DONE kehren Sie wieder zum Dialogfeld PREFERENCES zurück. Ihre Lizenz wird wie in Abbildung 1.5 angezeigt.

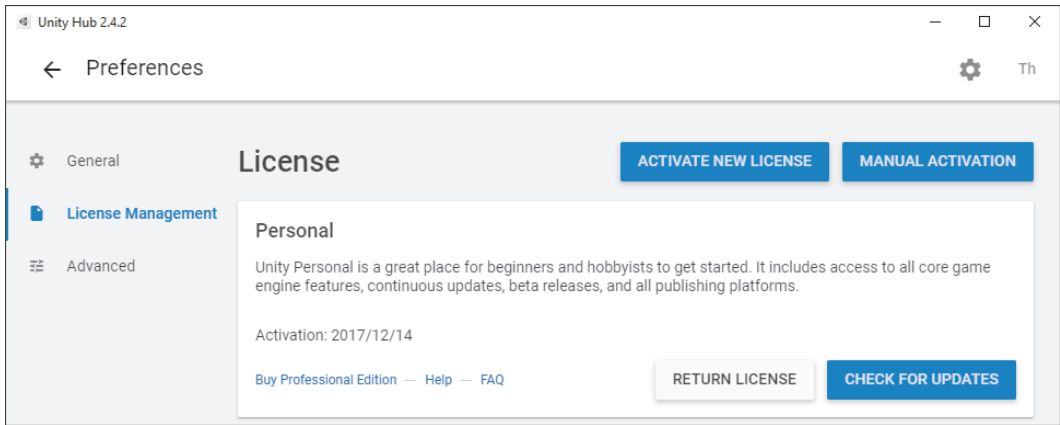


Abbildung 1.5 Lizenz »Personal«

Verlassen Sie das Dialogfeld PREFERENCES über den Pfeil oben links. Sie gelangen zur Hauptseite des Dialogfelds UNITY HUB. In Abbildung 1.6 sehen Sie die Version 2.4.2. Sobald eine neue Version oder ein neues Release (= Unterversion) des Unity Hubs erscheint, wird Ihnen ein Update vorgeschlagen. Ich empfehle, dem Vorschlag zu folgen.

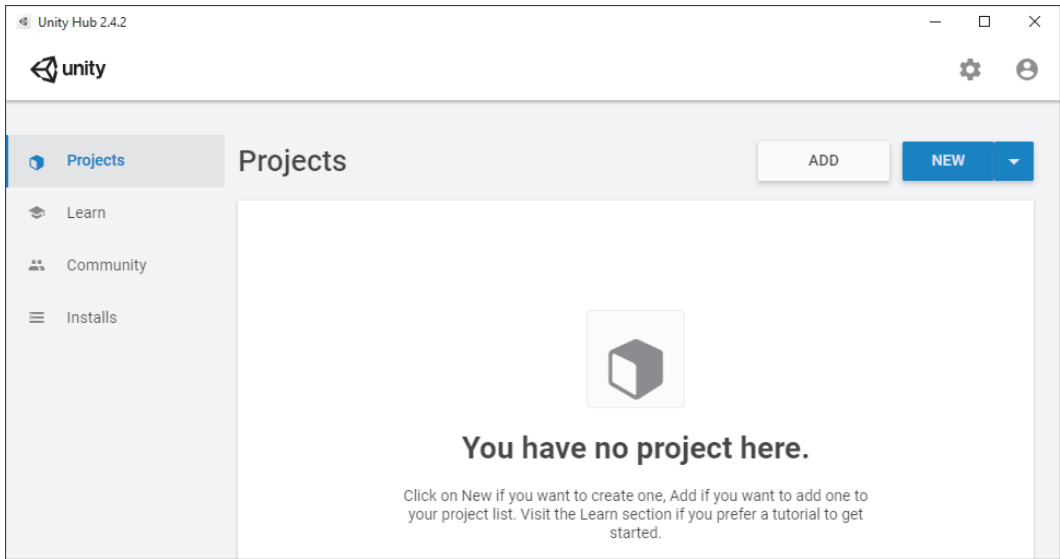


Abbildung 1.6 Unity Hub, Projects

Auf der linken Seite befindet sich ein Menü, mit dessen Hilfe Sie verschiedene Seiten aufrufen können. Auf der Seite PROJECTS werden später Ihre eigenen Unity-Projekte angezeigt.

Neue Versionen, neue Releases und neue Vorversionen von Unity erscheinen parallel und in kurzen Zeitabständen. Alle werden ständig weiterentwickelt und können von Ihnen gleichzeitig genutzt werden. So können Sie permanent von der Modernisierung von Unity profitieren. Der Unity Hub ist dafür vorgesehen, Ihnen die parallele Nutzung und den schnellen Umstieg zu ermöglichen.

Auf der Seite INSTALLS werden später die verschiedenen Unity-Versionen angezeigt, die Sie installiert haben, siehe Abbildung 1.7.

Parallele Nutzung

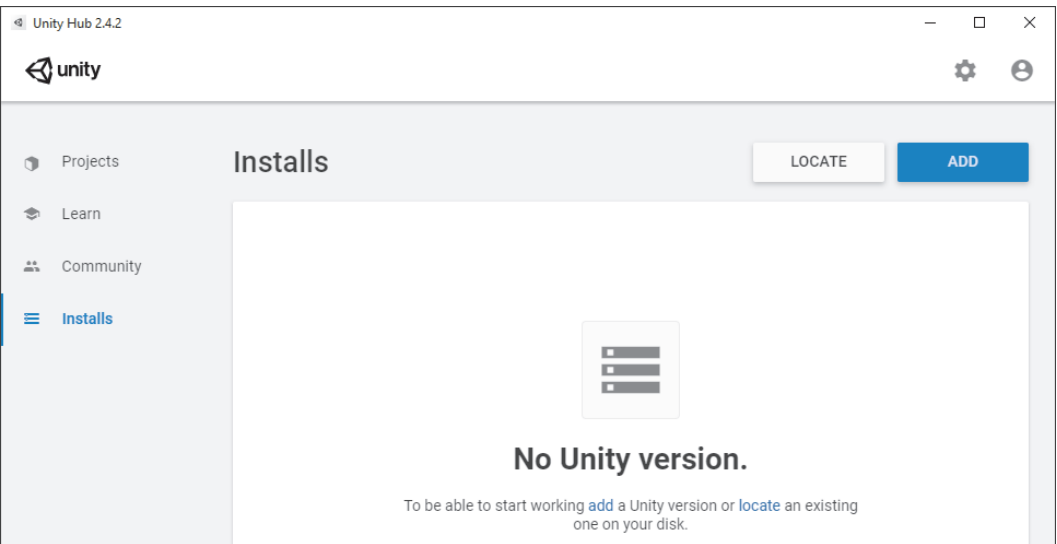


Abbildung 1.7 Unity Hub, Installs

1.5 Unity-Version installieren

Zur Installation einer Unity-Version betätigen Sie auf der Seite INSTALLS die Schaltfläche ADD, siehe Abbildung 1.7. Es erscheint die erste Seite des Dialogfelds ADD UNITY VERSION, siehe Abbildung 1.8. Es gibt ein empfohlenes Release mit Langzeitunterstützung (engl.: Long Term Support, kurz LTS), offizielle Releases und Vor-Releases. Wählen Sie z. B. das Vor-Release Beta 2 zu Unity 2021.1.0 aus, und betätigen Sie die Schaltfläche NEXT.

Version auswählen

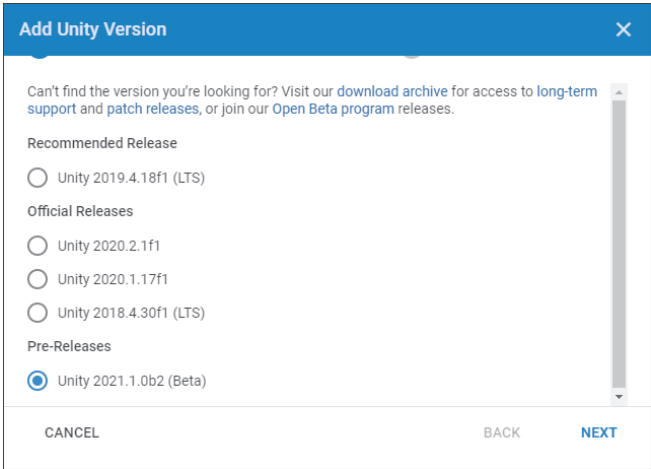


Abbildung 1.8 Unity-Version hinzufügen

Komponenten
hinzufügen

Anschließend erscheint die zweite Seite des Dialogfelds ADD UNITY VERSION, siehe Abbildung 1.9. Hier können Sie weitere Komponenten zur Installation auswählen, u. a. eine Version von *Visual Studio* der Firma Microsoft. Alternativ können Sie *MonoDevelop* installieren, wie es in Abschnitt 1.6, »MonoDevelop installieren«, beschrieben wird.

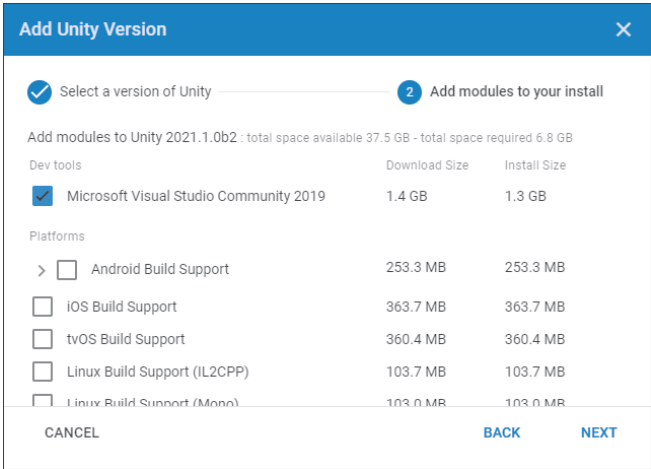


Abbildung 1.9 Weitere Komponenten

Weitere Komponenten werden zurzeit noch nicht benötigt. Sie könnten im Übrigen jederzeit nachinstalliert werden. Daher wird als Nächstes die Schaltfläche DONE betätigt.

Die eigentliche Installation startet und nimmt eine gewisse Zeit in Anspruch. Es erscheint ein neues Element mit einem Fortschrittsbalken am oberen Rand, an dem Sie den Stand der Installation erkennen können, siehe Abbildung 1.10. Über das Menü mit den drei Punkten könnten Sie eine laufende Installation jederzeit abbrechen.

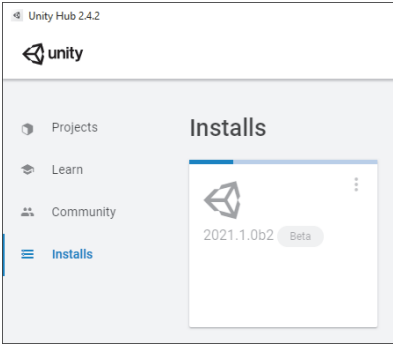


Abbildung 1.10 Unity Hub, laufende Installation

Haben Sie *Visual Studio* als Zusatzmodul ausgewählt, gilt Folgendes:

Visual Studio

- ▶ Nach einiger Zeit erscheint das Dialogfeld END USER LICENSE AGREEMENT für Visual Studio, in dem die allgemeinen Lizenzbedingungen akzeptiert werden müssen.
- ▶ Nach der Fertigstellung der Unity-Installation sollte der Unity Hub beendet und ein Neustart des Rechners durchgeführt werden.
- ▶ Sie nehmen in Ihrem ersten Unity-Projekt eine weitere notwendige Einstellung vor, damit Visual Studio anschließend in Ihren sämtlichen Unity-Projekten automatisch genutzt wird, siehe Abschnitt 4.3.1, »Wählen Sie die Entwicklungsumgebung«.

Im Folgenden gehe ich davon aus, dass Sie eine Unity-Version installiert haben.

1.6 MonoDevelop installieren

MonoDevelop kann unter allen Unity-Versionen installiert und genutzt werden. Rufen Sie zum Download von MonoDevelop in Ihrem Browser die folgende Seite auf:

<https://unity3d.com/de/get-unity/download/archive>

Wählen Sie den Reiter 2017.X aus. Die im Dezember 2020 aktuelle 2017er-Version (hier: 2017.4.40) steht oben in der Liste. Wählen Sie in der Liste DOWNLOADS (WIN) das UNITY-INSTALLATIONSPROGRAMM aus, siehe Abbildung 1.11.

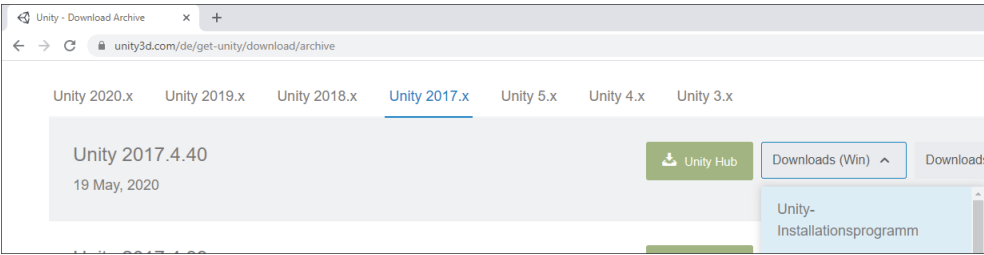


Abbildung 1.11 Unity-Website, Version 2017.4.40 herunterladen

Es wird die Installationsdatei *UnityDownloadAssistant-2017.4.40f1.exe* heruntergeladen. Starten Sie die Installation. Wählen Sie unter den Komponenten ausschließlich MONODEVELOP aus, siehe Abbildung 1.12.

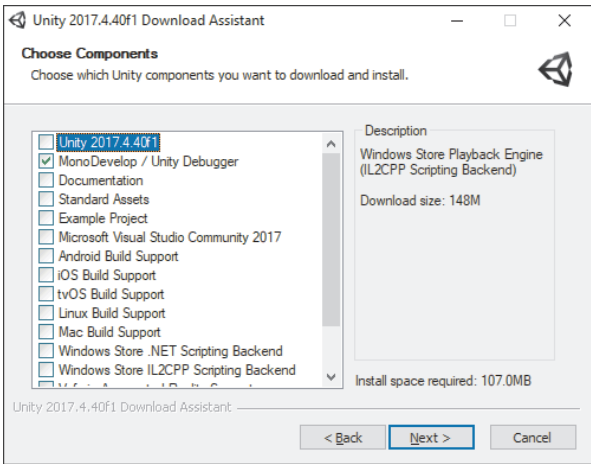


Abbildung 1.12 Komponente MonoDevelop

Die Unity-Version 2017.4.40 selbst muss nicht installiert werden. Sie können als Installationsverzeichnis ebenfalls *C:\UnityHub* auswählen.

In Ihrem ersten Unity-Projekt nehmen Sie eine weitere notwendige Einstellung vor, damit MonoDevelop anschließend in Ihren sämtlichen Unity-Projekten automatisch genutzt wird, siehe Abschnitt 4.3.1, »Wählen Sie die Entwicklungsumgebung«.

1.7 Beispielprojekte und Assets

In diesem Buch arbeite ich mit einer Reihe von Beispielprojekten, die Ihnen sowohl als Hilfestellung bei der eigenen Entwicklung als auch zum Kennenlernen und Ausprobieren in ausführbarer Form dienen. Sie stehen in den Materialien zum Buch über die Webseite <https://www.rheinwerk-verlag.de/5293> zum Download zur Verfügung.

Bei Unity wird jedes Projekt in einem eigenen Projektverzeichnis gespeichert. Legen Sie zur besseren Übersicht ein eigenes Oberverzeichnis an, z. B. *C:\UnityProjekte*. Kopieren Sie alle Verzeichnisse mit meinen Beispielprojekten nach dem Download in dieses Verzeichnis.

In den Beispielprojekten nutze ich eine Reihe von einfachen vorgefertigten Elementen, sogenannte *Assets*. Sie finden sie, zusammen mit den bereits erwähnten Projektverzeichnissen, im Verzeichnis *FreieAssets* in den Materialien zum Buch.

In Abschnitt 22.6, »Bonusprojekte«, werden zwei Bonusprojekte beschrieben, die sich ebenfalls in den Materialien zum Buch befinden. Sie sind den bekannten Spielen *Pacman* und *Frogger* nachempfunden. Mit den Fähigkeiten, die Sie aus diesem Buch erworben haben, sind Sie in der Lage, den Aufbau dieser Projekte zu erkennen und sie mit eigenen Ideen selbstständig weiterzuentwickeln.

1.8 Unity-Projekte und Unity-Versionen

Aufgrund der kurzen Zeitabstände beim Erscheinen neuer Unity-Versionen kommt es häufig vor, dass die Entwicklung eines Unity-Projekts zunächst unter einer bestimmten Version beginnt und später unter einer anderen Version fortgesetzt wird. Beim Öffnen eines Projekts in einer neueren Version ist ein Upgrade notwendig. Mithilfe des Unity Hubs ist das leicht möglich.

Beim Öffnen eines Projekts wird zunächst die gewünschte Version ausgewählt. Klicken Sie dazu auf den kleinen Pfeil in der Spalte Unity-Version. Es erscheint eine Liste der installierten Unity-Versionen zur Auswahl. In Abbildung 1.13 wird das am Beispiel des Projekts *TomsJumpAndRun* aus Kapitel 3, »Spielen Sie ein 2D-Jump&Run-Spiel«, gezeigt. Es wurde zunächst unter der Unity-Version 2019.4.14f1 entwickelt und soll unter der Unity-Version 2020.1.12f1 weiterentwickelt werden.

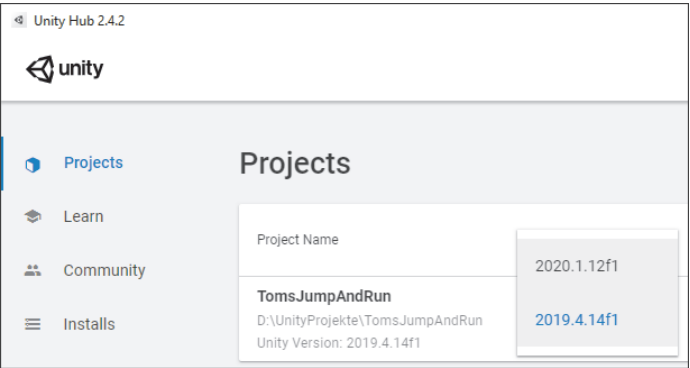


Abbildung 1.13 Auswahl einer Unity-Version

Nach Auswahl der Unity-Version wird das Projekt geöffnet, indem die gesamte Projektzeile angeklickt wird. Es erscheint das in Abbildung 1.14 dargestellte Dialogfeld. Die Konvertierung nimmt eine gewisse Zeit in Anspruch. Anschließend kann das Unity-Projekt weiterbearbeitet werden.

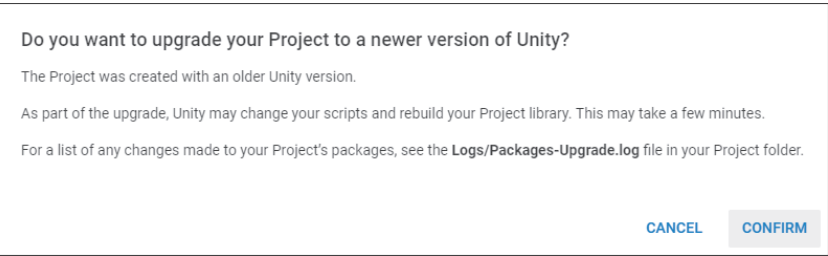


Abbildung 1.14 Upgrade eines Projekts

Die Beispielprojekte, die Screenshots und die Erläuterungen in diesem Buch sind mithilfe der Unity-Version 2021.1.0b2 entstanden, also der zweiten Beta-Vorversion der Unity-Version 2021.1.0.

Die Erfahrungen der letzten Jahre zeigen, dass in neuere Unity-Versionen zwar Verbesserungen einfließen, aber die grundsätzliche Orientierung erhalten bleibt. Daher werden die Inhalte dieses Buchs Sie noch sehr lange bei der Entwicklung von Unity-Projekten unterstützen.

Kapitel 9

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 2, »Das erste 2D-Projekt«, 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

9.1 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.

9.1.1 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 vorn 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 z. B. 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 und dient 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.

Hinweis

Es gibt u. a. 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.
- ▶ 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 Glühlampe.
- ▶ 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.

9.1.2 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. Klappen Sie dazu jeweils das Menü der HIERARCHY VIEW mithilfe eines Klicks auf das Pluszeichen auf:

- Würfel** ▶ Erzeugen Sie einen Würfel mithilfe des Menüpunkts 3D OBJECT • CUBE. Er verbleibt in der Mitte, und zwar an der Position 0/0/0.
- Kugel** ▶ Erzeugen Sie als Nächstes mithilfe des Menüpunkts 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.

- ▶ Es folgt eine Kapsel, die mithilfe des Menüpunkts 3D OBJECT • CAPSULE erstellt wird. Sie wird auf der positiven X-Achse verschoben, und zwar an die Position 4/0/0. **Kapsel**
- ▶ Als Letztes erstellen Sie über den Menüpunkt 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 9.1.

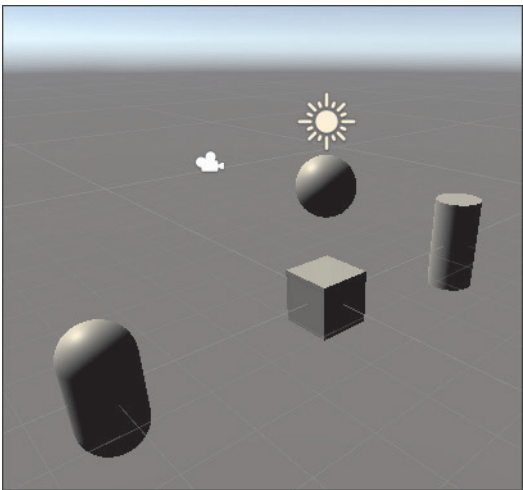


Abbildung 9.1 Einfache 3D-Objekte in der »Scene View«

Hinweis

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.

9.1.3 Farbiges Oberflächenmaterial

Die Oberflächen der 3D-Objekte sollen eine Farbe erhalten. Erstellen Sie das Asset-Verzeichnis MATERIALS. Öffnen Sie durch einen Klick auf das Pluszeichen das Menü der PROJECT VIEW, und erzeugen Sie in dem neuen Verzeichnis mithilfe des Menüpunkts MATERIAL ein neues Asset für Oberflächenmaterialien. Nennen Sie es OrangeMat als Abkürzung für *Orangefarbenes Oberflächenmaterial*.

Neues Material

Albedo 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 9.2) die RGB-Werte 255/128/0. Die Eigenschaft ALBEDO bestimmt das Reflexionsverhalten von Oberflächen, die angeleuchtet werden.

Erstellen Sie durch Kopie ein weiteres Asset für Oberflächenmaterialien, und nennen Sie es HellblauMat. Es erhält für die Eigenschaft ALBEDO die RGB-Werte 0/128/255.

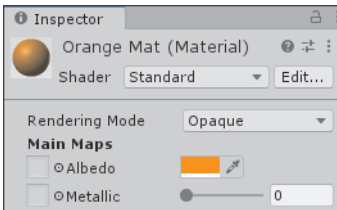


Abbildung 9.2 Wichtigste Farbe für ein Oberflächenmaterial

Material zuordnen Zur Zuordnung der Oberflächenmaterialien klappen Sie bei den einzelnen Objekten in der Komponente MESH RENDERER den Bereich MATERIALS auf. Ziehen Sie das jeweilige Material-Asset auf die Eigenschaft ELEMENT 0. Die Kugel wird orangefarben, die Kapsel hellblau, siehe Abbildung 9.7.

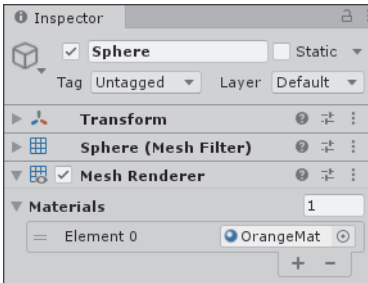


Abbildung 9.3 Oberflächenmaterial zuordnen

9.1.4 Oberflächenmaterial mit Textur

Eine Textur ist ein zweidimensionales Bild, das auf die Oberfläche eines dreidimensionalen Körpers gelegt wird, damit dieser realistischer wirkt. Erstellen Sie das Asset-Verzeichnis TEXTURES, und importieren Sie aus dem Verzeichnis *FreieAssets* die beiden Bilddateien *Holzfass.png* und *Mauer.png* in das neue Verzeichnis. Sie sollen als Texturen dienen.

Erstellen Sie im Asset-Verzeichnis MATERIALS ein neues Material, und nennen Sie es *HolzfassMat* als Abkürzung für *Oberflächenmaterial mit Holzfass-Textur*. Markieren Sie es, und wählen Sie in der Liste SHADER (dt.: Schattierung) statt des Eintrags STANDARD den Eintrag LEGACY SHADERS • DIFFUSE, siehe Abbildung 9.4.

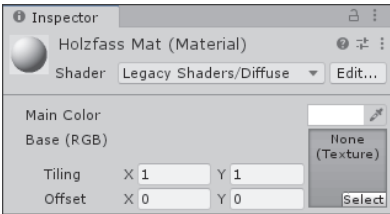


Abbildung 9.4 Typ der Schattierung einstellen

Bei diesem Typ der Schattierung haben Sie die Möglichkeit, nach Betätigung der Schaltfläche SELECT eine Textur einzustellen. Wählen Sie im Dialogfeld SELECT TEXTURE die zuvor importierte Textur HOLZFASS, siehe Abbildung 9.5.

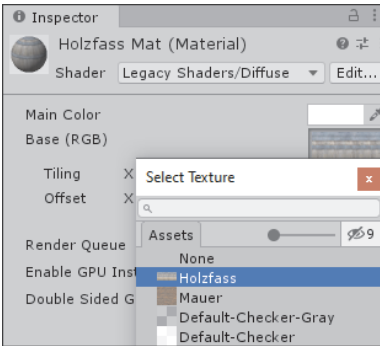


Abbildung 9.5 Textur auswählen

Erstellen Sie ein weiteres Material mit dem Namen *MauerMat*, und geben Sie ihm auf dieselbe Weise die Textur MAUER.

Hinweis

Standardmäßig wird ein Bild bei der Texturierung nur einmal auf einer Oberfläche abgebildet. Abhängig von der Größe der Oberfläche kann es aber realistischer wirken, wenn das Textur-Bild wie eine Wand- oder Bodenfliese mehrmals nebeneinander und übereinander abgebildet wird. Dieser Vorgang wird *Tiling* (engl.: to tile, dt.: Fliesenlegen) genannt.

Stellen Sie beim Material `MauerMat` den `TILING`-Wert für `X` auf 2, siehe Abbildung 9.6. In diesem Fall erscheint die Textur in `X`-Richtung zweimal nebeneinander auf der Oberfläche. Der Wert für `Y` bleibt beim Standardwert 1.

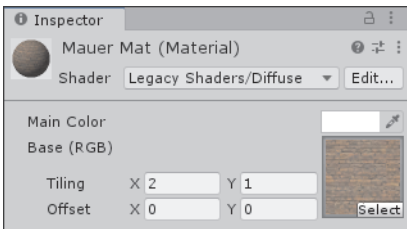


Abbildung 9.6 Tiling einstellen

Hinweis

Sind die Übergänge der neben- bzw. übereinanderliegenden Fliesen nicht sichtbar, ist eine Bilddatei besonders gut für das Tiling geeignet. Die Strukturen am linken und am rechten Rand eines Bilds sollten also ebenso zueinander passen wie diejenigen am oberen und am unteren Rand.

Anschließend sieht das Asset-Verzeichnis `MATERIALS` aus wie in Abbildung 9.7.

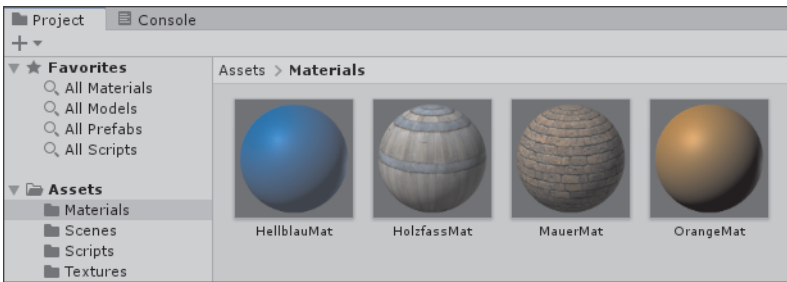


Abbildung 9.7 Vier Materialien

Weisen Sie dem Zylinder das Material `HolzfassMat` und dem Würfel das Material `MauerMat` zu.

9.1.5 Oberflächenmaterial wechseln

Sie können das Aussehen von 3D-Objekten zur Laufzeit eines Projekts verändern, indem Sie z. B. das Oberflächenmaterial wechseln, wie im nachfolgenden Code für den Zylinder:

Klasse »Cylinder«

```
using UnityEngine;
public class Cylinder : MonoBehaviour
{
    public Material hellblauMat;
    public Material holzfassMat;

    void Update()
    {
        if (Input.GetKeyDown (KeyCode.B))
            GetComponent<MeshRenderer>().material = hellblauMat;
        else if (Input.GetKeyDown (KeyCode.F))
            GetComponent<MeshRenderer>().material = holzfassMat;
    }
}
```

Listing 9.1 Klasse »Cylinder«

Es werden die beiden öffentlich zugänglichen Variablen `hellblauMat` und `holzfassMat` vom Datentyp `Material` erstellt. Falls der Benutzer die Taste `[B]` oder die Taste `[F]` betätigt, erhält die Eigenschaft `material` der Komponente `MESH RENDERER` des zugehörigen Spielobjekts einen neuen Wert.

Mesh Renderer

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

Material zuordnen

9.1.6 Ansicht in der »Scene View« gestalten

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

Handsymbol

- *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 Verschieben der Maus bei gedrückter rechter Maustaste nach links, rechts, oben oder unten.
- *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.

Verschieben

Drehen

Zoomen

Gizmo Rechts oben in der SCENE VIEW befindet sich ein kleines Hilfselement, ein sogenanntes *Gizmo* (siehe Abbildung 9.8). Es zeigt die aktuelle Lage der drei Koordinatenachsen und die aktuelle Darstellungsart.

Hinweise

Wie in Abschnitt 2.8.1, »Die Eigenschaften der Transform-Komponente«, 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 vorn gestreckt. Damit weist er in Richtung der positiven Z-Achse.

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.

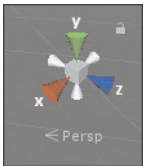


Abbildung 9.8 Gizmo

- Isometrisch** Klicken Sie einmal auf den Schriftzug **PERSP** (Abkürzung für *perspektivisch*) unter dem Gizmo: Die Darstellungsart wechselt zu *isometrisch*, der Text des Schriftzugs auf **ISO**. Alle 3D-Objekte werden unabhängig von der Entfernung zum Betrachter gleich groß dargestellt. Der räumliche Effekt geht verloren. Ein erneuter Klick auf den Schriftzug 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 durch Verschieben, Drehen und Zoomen ungefähr die Ansicht in Abbildung 9.9 herzustellen: Die positive X-Achse weist nach rechts, die positive Y-Achse weist nach oben. Die Ansicht wird gedreht,

sodass die linke und die obere Seite des Würfels zu sehen sind. Das ist beim ersten Mal nicht einfach, stellt aber eine gute Übung dar.

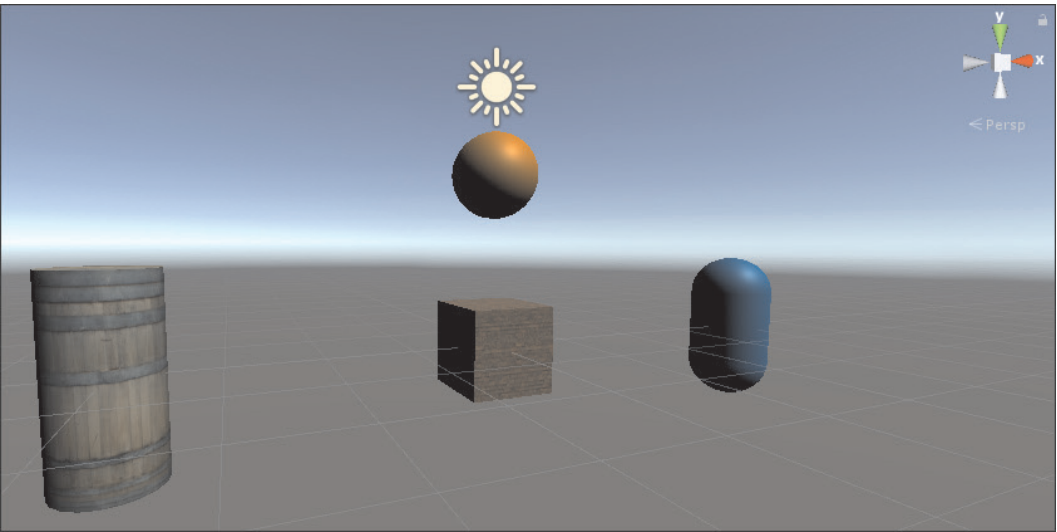


Abbildung 9.9 Gewünschte Ansicht

Bei Bedarf können Sie nun auch die Ansicht in der GAME VIEW gleichzeitig 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

9.2 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*.

9.2.1 Spielobjekte drehen

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

Hinweis

Was ist ein Drehmoment? Nehmen wir an, die vorhandenen 3D-Objekte könnten sich nur drehen, aber ihre Position nicht verändern. Nehmen wir des Weiteren an, Sie stecken eine lange Stange durch ein 3D-Objekt, z. B. durch den Würfel, der im Zentrum steht, und zwar genau entlang der Y-Achse. Drücken Sie in X-Richtung außerhalb des Würfels gegen die Stange, 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 9.2 Klasse »Cube«

Betätigt der Benutzer die Taste **U** oder die Taste **I**, 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).

Hinweis

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.

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

Es sollen zwei weitere Drehungen hinzukommen. Ergänzen Sie zunächst die Methode Update() der Klasse Cylinder wie folgt, damit eine Drehung des Zylinders um die X-Achse ermöglicht wird:

```
void Update()
{
    ...
    else if (Input.GetKeyDown(KeyCode.Y))
        GetComponent<Rigidbody>().AddTorque(5, 0, 0);
    else if (Input.GetKeyDown(KeyCode.X))
        GetComponent<Rigidbody>().AddTorque(-5, 0, 0);
}
```

Listing 9.3 Klasse »Cylinder«, Ergänzung

Erstellen Sie wie folgt die Klasse Capsule, damit eine Drehung der Kapsel um die Z-Achse ermöglicht wird:

```
using UnityEngine;
public class Capsule : MonoBehaviour
{

```



```

void Update()
{
    if (Input.GetKeyDown (KeyCode.A))
        GetComponent<Rigidbody>().AddTorque (0, 0, 5);
    else if (Input.GetKeyDown (KeyCode.S))
        GetComponent<Rigidbody>().AddTorque (0, 0, -5);
}

```

Listing 9.4 Klasse »Capsule«

Testen 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 9.10 sehen Sie eine Momentaufnahme, nachdem alle drei Objekte in Drehung versetzt wurden.

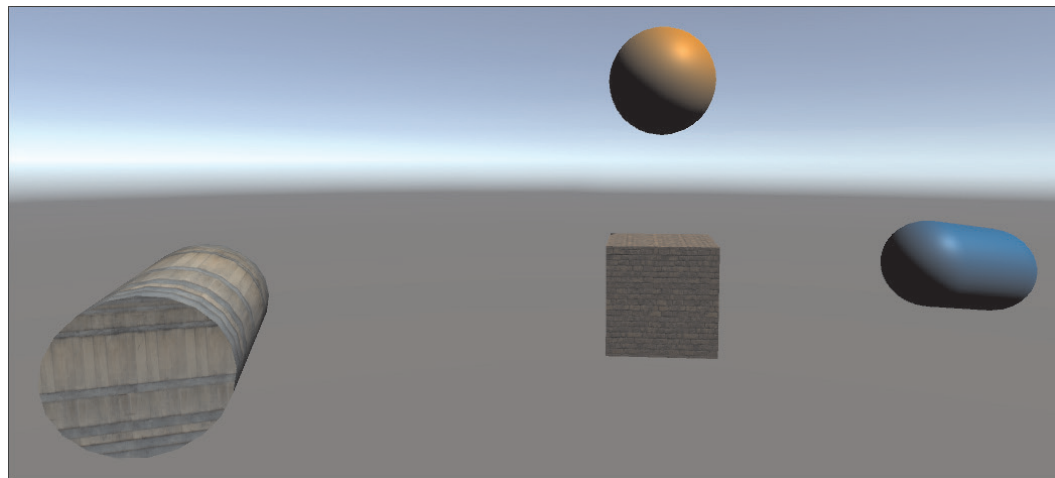


Abbildung 9.10 Zylinder, Würfel und Kapsel drehen sich

9.2.2 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. Erstellen Sie die Klasse Sphere mit folgendem Code:

```

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 (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 9.5 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

Gesamtzeit und
Gesamtstrecke

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.

Anteilige
Verschiebung

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.

Testen

Ordnen Sie das C#-Script dem Spielobjekt `Sphere` zu, und testen Sie die Bewegung. Da das Spielobjekt keinen `Rigidbody` besitzt, bewegt es sich durch das Spielobjekt `Cube` hindurch (siehe Abbildung 9.11). Das gilt auch, wenn sich das Spielobjekt `Cube` zurzeit drehen sollte. Stellen Sie unterschiedliche Zielpunkte und Laufzeiten ein, und starten Sie die Bewegung erneut. Versehen Sie das Spielobjekt `Sphere` kurzzeitig mit einem `Rigidbody`, und beobachten Sie die Auswirkungen.

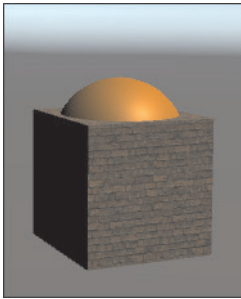


Abbildung 9.11 Kugel bewegt sich von A nach B

9.2.3 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 selbst gedreht.
- ▶ Die Kamera wird um einen bestimmten Punkt gedreht.

In Kapitel 15, »Jagen auf einem 3D-Terrain«, finden Sie außerdem ein Projekt, bei dem der Benutzer selbst mit der Kamera herauszoomen und wieder hineinzoomen kann.

Zoomen

Es folgt der Code der Klasse `KameraBewegen`:

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 9.6 Klasse »KameraBewegen«

Die Klasse erhält nicht den Namen `Camera`, weil es bereits eine gleichnamige Komponente des Spielobjekts `Main Camera` gibt.

Name der Klasse

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.

`Translate()`

Rotate() 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.

RotateAround() 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:

- ▶ 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

Testen Das C#-Script wird 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 9.12 sehen Sie die Ansicht, nachdem die Taste **[J]** mehrmals betätigt wurde.

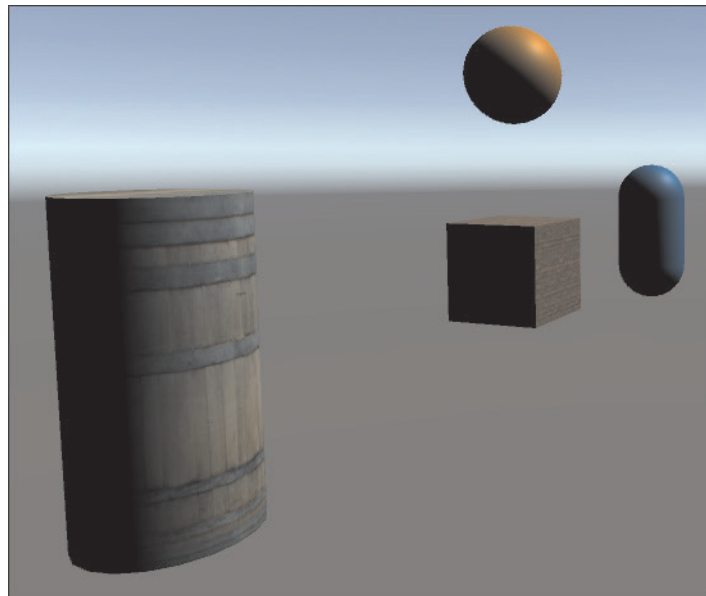


Abbildung 9.12 Kamera wird um Zylinder gedreht

9.2.4 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 bestimmte Achse drehen, bis ein Zielwinkel B erreicht ist. Das wird mithilfe des nachfolgenden Codes für das Spielobjekt `Main Camera` realisiert:

Klasse
»KameraBewegen«

```
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 9.7 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 9.2.2, »Animiert verschieben«. Zunächst werden einige Variablen deklariert, die innerhalb der gesamten Klasse gelten. Sie werden innerhalb der Methoden erläutert.

Gesamtzeit und
Gesamtwinkel

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).

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

Anteilige Drehung

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.

Drehwinkel

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.

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

Testen

9.2.5 Übersicht

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

Alle Tastencodes

Taste	Erläuterung
B	Zylinder erhält Material mit hellblauer Farbe.
F	Zylinder erhält Material mit Holzfass-Textur.
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 9.1 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	Einführung	17
2	Das erste 2D-Projekt	29
3	Spielen Sie ein 2D-Jump&Run-Spiel	45
4	Entwickeln Sie ein 2D-Jump&Run-Spiel	54
5	Ein 2D-Breakout-Spiel	105
6	Ein 2D-Spiel für zwei Spieler	143
7	Ein Gedächtnistrainer als 2D-Projekt	163
8	Ein 2D-Space-Shooter	183
9	Das erste 3D-Projekt	205
10	Eine 3D-Animation	225
11	Ein 3D-Balancer	239
12	Ein 3D-Tetris	251
13	Ein Kopfrechentrainer als 3D-Projekt	264
14	Golf spielen auf einem 3D-Terrain	277
15	Jagen auf einem 3D-Terrain	302
16	Eine Schlange aus 3D-Joints	325
17	Ein Renntraining und ein Autorennen	339
18	Erkunden Sie das Verlies	370
19	Ein Programmierkurs in C#	404
20	Speichern Sie eine Highscore-Liste	444
21	Arbeiten Sie mit mehreren Szenen	455
22	Allgemeine Hinweise	462

Inhalt

1	Einführung	17
1.1	Was machen wir mit Unity?	17
1.2	Wie entsteht der programmierte Spielablauf?	18
1.3	Dateiendungen anzeigen lassen	19
1.4	Unity Hub installieren	19
1.5	Unity-Version installieren	23
1.6	MonoDevelop installieren	25
1.7	Beispielprojekte und Assets	27
1.8	Unity-Projekte und Unity-Versionen	27
2	Das erste 2D-Projekt	29
2.1	Erstellung eines neuen Projekts	29
2.2	Wichtige Bereiche im Unity Editor	30
2.3	Das Spielobjekt »Main Camera«	32
2.4	Assets importieren	33
2.5	Spielobjekte einfügen	34
2.6	Ändern der Hierarchie	35
2.7	Eine Szene speichern	37
2.8	Die Komponente »Transform«	38
2.8.1	Die Eigenschaften der Transform-Komponente	38
2.8.2	Werte in der »Inspector View« ändern	40
2.9	Die Ansicht in der »Scene View«	41
2.9.1	Positionswerte mithilfe der Maus ändern	42
2.9.2	Rotationswerte mithilfe der Maus ändern	43
2.9.3	Scale-Werte mithilfe der Maus ändern	44

3	Spieren Sie ein 2D-Jump&Run-Spiel	45
3.1	Wie geht das Spiel?	45
3.2	Unsere ersten Unity-Elemente	47
3.2.1	Assets	48
3.2.2	Spielobjekte	49
4	Entwickeln Sie ein 2D-Jump&Run-Spiel	54
4.1	Erzeugen Sie Projekt und Assets	54
4.2	Fügen Sie Spielobjekte ein	55
4.2.1	Schaffen Sie einen Hintergrund	55
4.2.2	Erzeugen Sie das Spielfeld	56
4.2.3	Setzen Sie den Affen auf den Boden	58
4.3	Erstellen Sie den Spielablauf	61
4.3.1	Wählen Sie die Entwicklungsumgebung	61
4.3.2	Führen Sie die Klasse »Spieler« ein	62
4.3.3	Bewegen Sie den Affen	66
4.3.4	Verhindern Sie die Drehung	71
4.3.5	Begrenzen Sie die Bewegung zur Seite	71
4.3.6	Treffen Sie die Bananen	73
4.3.7	Meiden Sie die Tiger	77
4.3.8	Die geschweiften Klammern	79
4.3.9	Die Tiger bewegen sich	80
4.4	Gestalten Sie die Benutzeroberfläche	82
4.4.1	Erstellen Sie die erste Anzeige	82
4.4.2	Sammeln Sie Punkte	84
4.4.3	Verlieren Sie Leben	87
4.4.4	Messen Sie die Spielzeit	88
4.4.5	Speichern Sie Werte dauerhaft	91
4.4.6	Geben Sie den Benutzern Hinweise	93
4.4.7	Starten Sie ein neues Spiel	97
4.4.8	Beenden Sie die Anwendung	100
4.4.9	Ideen für Ihre Erweiterungen	101
4.5	Erzeugen Sie eine ausführbare Version	102

5	Ein 2D-Breakout-Spiel	105
5.1	Führen Sie das Spiel aus	105
5.2	Erzeugen Sie Projekt und Assets	106
5.2.1	Fügen Sie ein Audio-Asset ein	107
5.2.2	Erstellen Sie ein 2D-Material	107
5.2.3	Lernen Sie 2D-Materialien kennen	108
5.2.4	Erzeugen Sie ein Prefab	110
5.3	Fügen Sie Spielobjekte ein	111
5.3.1	Füllen Sie das Spielfeld	111
5.3.2	Erzeugen Sie einen Ziegel	113
5.3.3	Wiederholen Sie den Vorgang	114
5.3.4	Wiederholen Sie die Wiederholung	116
5.3.5	Erzeugen Sie unterschiedliche Ziegel	117
5.3.6	Mehrfache Verzweigung mit »switch«-Ausdruck	119
5.4	Erstellen Sie den Spielablauf	119
5.4.1	Senden Sie den Ball ab	119
5.4.2	Bewegen Sie den Spieler	122
5.4.3	Sammeln Sie Punkte	123
5.4.4	Verlieren Sie Leben	126
5.5	Gestalten Sie die Benutzeroberfläche	129
5.5.1	Exportieren Sie ein Unity Package	129
5.5.2	Importieren Sie ein Unity Package	130
5.5.3	Passen Sie die Benutzeroberfläche an	131
5.5.4	Punkte, Leben und Infos anzeigen	133
5.5.5	Messen Sie die Spielzeit	134
5.5.6	Zeigen Sie die vorherige Zeit an	136
5.5.7	Starten Sie ein neues Spiel	137
5.5.8	Beenden Sie die Anwendung	141
5.5.9	Ideen für Ihre Erweiterungen	142
6	Ein 2D-Spiel für zwei Spieler	143
6.1	Führen Sie das Spiel aus	143
6.2	Bereiten Sie das Spiel vor	145
6.2.1	Erzeugen Sie Projekt und Assets	146

6.2.2	Erzeugen Sie Spielfeld und UI	146
6.2.3	Gestalten Sie das Spielfeld	148
6.3	Erstellen Sie den Spielablauf	151
6.3.1	Führen Sie den Aufschlag aus	151
6.3.2	Bewegen Sie die Spieler vertikal	153
6.3.3	Bewegen Sie die Spieler horizontal	156
6.3.4	Sammeln Sie Punkte	158
6.3.5	Eine kleine Übung	160
6.3.6	Ideen für Ihre Erweiterungen	161
6.4	Künstliche Intelligenz	161
7	Ein Gedächtnistrainer als 2D-Projekt	163
7.1	Führen Sie das Training aus	163
7.2	Bereiten Sie das Training vor	164
7.3	Das Training für drei Zahlen	165
7.3.1	Verteilen Sie die Zahlen	166
7.3.2	Vermeiden Sie doppelte Positionen	169
7.3.3	Löschen Sie die Zahlen	171
7.3.4	Prüfen Sie die Reihenfolge	172
7.4	Die Erweiterung des Trainings	174
7.4.1	Machen Sie das Training leichter	175
7.4.2	Machen Sie das Training schwerer	176
7.4.3	Optimieren Sie das Training	179
7.4.4	Ideen für Ihre Erweiterungen	181
8	Ein 2D-Space-Shooter	183
8.1	Bereiten Sie das Spiel vor	184
8.1.1	Gestalten Sie die beiden Explosions-Prefabs	185
8.1.2	Erzeugen Sie Ihr Raumschiff und die Geschosse	186
8.1.3	Erstellen Sie die anderen Raumschiffe	188
8.1.4	Gestalten Sie die Energieanzeige	189
8.1.5	Erstellen Sie die Benutzeroberfläche	189

8.2	Erstellen Sie den Spielablauf	190
8.2.1	Bewegen Sie Ihr Raumschiff, und feuern Sie	190
8.2.2	Bewegen Sie die Geschosse nach dem Abfeuern	193
8.2.3	Bewegen Sie die anderen Raumschiffe	193
8.2.4	Lassen Sie die Raumschiffe explodieren	195
8.2.5	Kollidieren Sie mit den anderen Raumschiffen	198
8.2.6	Führen Sie weitere Änderungen der Energie herbei	199
8.2.7	Messen Sie die Zeit, und beenden Sie das Spiel	201
8.2.8	Ausführbare Version	203
8.2.9	Eine kleine Übung	203
8.2.10	Ideen für Ihre Erweiterungen	204
9	Das erste 3D-Projekt	205
9.1	Grundlagen eines 3D-Projekts	205
9.1.1	Kamera, Skybox und Licht	205
9.1.2	Einfache 3D-Objekte	206
9.1.3	Farbiges Oberflächenmaterial	207
9.1.4	Oberflächenmaterial mit Textur	208
9.1.5	Oberflächenmaterial wechseln	210
9.1.6	Ansicht in der »Scene View« gestalten	211
9.2	Verschieben und Drehen	213
9.2.1	Spielobjekte drehen	213
9.2.2	Animiert verschieben	216
9.2.3	Kamera bewegen	218
9.2.4	Animiert drehen	221
9.2.5	Übersicht	223
10	Eine 3D-Animation	225
10.1	Schaffen Sie die Voraussetzungen	225
10.1.1	Betrachten Sie die fertige Animation	225
10.1.2	Bauen Sie das Beispiel auf	226
10.2	Erstellen Sie die Animation	228
10.2.1	Legen Sie die Animation an	228

10.2.2	Drehen Sie das rechte Bein	229
10.2.3	Erstellen Sie weitere Keyframes	230
10.2.4	Stellen Sie die Keyframes ein	231
10.2.5	Verschieben Sie das rechte Bein	232
10.3	Arbeiten Sie mit dem »Animator Controller«	233
10.3.1	Gestalten Sie die States	233
10.3.2	Erstellen Sie die Parameter	234
10.3.3	Erzeugen Sie die Transitions	235
10.4	Fügen Sie das C#-Script hinzu	236
10.4.1	Verbinden Sie Bewegung und Animation	236
10.4.2	Vervollständigen Sie die Animation	237
10.4.3	Ideen für Ihre Erweiterungen	238
11	Ein 3D-Balancer	239
11.1	Führen Sie das Spiel aus	239
11.2	Bereiten Sie das Spiel vor	240
11.2.1	Erzeugen Sie Projekt und Assets	240
11.2.2	Erzeugen Sie Spielfeld und UI	241
11.2.3	Relative Transform-Werte	242
11.3	Erstellen Sie den Spielablauf	244
11.3.1	Drehen Sie die Platte	244
11.3.2	Bewegen Sie den Lederball und die Kamera	247
11.3.3	Ändern Sie die Punktzahl	248
11.3.4	Ideen für Ihre Erweiterungen	250
12	Ein 3D-Tetris	251
12.1	Führen Sie das Spiel aus	251
12.2	Bereiten Sie das Spiel vor	252
12.2.1	Erzeugen Sie Projekt und Assets	252
12.2.2	Erzeugen Sie Spielfeld und UI	253
12.2.3	Erstellen Sie das Würfel-Prefab	254

12.3	Erstellen Sie den Spielablauf	255
12.3.1	Bewegen Sie die Würfel	255
12.3.2	Eine »generische Liste«	256
12.3.3	Fügen Sie Elemente zur Liste hinzu	257
12.3.4	Entfernen Sie Elemente aus der Liste	260
12.3.5	Eine kleine Übung	263
12.3.6	Ideen für Ihre Erweiterungen	263
13	Ein Kopfrechentrainer als 3D-Projekt	264
13.1	Führen Sie das Training aus	264
13.2	Bereiten Sie das Training vor	265
13.3	Erstellen Sie den Trainingsablauf	266
13.3.1	Erzeugen Sie die Aufgabe und die Lösungen	266
13.3.2	Mischen Sie die Lösungen	269
13.3.3	Sammeln Sie Punkte	271
13.3.4	Verlieren Sie Leben	273
13.3.5	Ideen für Ihre Erweiterungen	275
14	Golf spielen auf einem 3D-Terrain	277
14.1	Führen Sie das Spiel aus	277
14.2	Bereiten Sie das Spiel vor	278
14.2.1	Erzeugen Sie Projekt und Landschaft	279
14.2.2	Weisen Sie der Landschaft eine Textur zu	279
14.2.3	Erstellen Sie die drei Ebenen	280
14.2.4	Fügen Sie den Rand hinzu	283
14.2.5	Erstellen Sie die beiden Rampen	284
14.2.6	Setzen Sie Spieler und Ziel in die Landschaft	286
14.2.7	Arbeiten Sie mit einem »Physic Material«	287
14.3	Erstellen Sie den Spielablauf	288
14.3.1	Schlagen Sie den Spielball	288
14.3.2	Versetzen Sie das Ziel	291

14.3.3	Vermeiden Sie den Verlust des Spielballs	292
14.3.4	Ideen für Ihre Erweiterungen	294
14.4	Ein weiteres Terrain	294
14.4.1	Erzeugen Sie zehn Ebenen	295
14.4.2	Fügen Sie den linken und den rechten Rand hinzu	296
14.4.3	Fügen Sie den unteren und den oberen Rand hinzu	297
14.4.4	Erzeugen Sie die erste Rampe	298
14.4.5	Erstellen Sie alle Rampen links	299
14.4.6	Erstellen Sie alle Rampen rechts	299
14.4.7	Setzen Sie die Positionen	300
15	Jagen auf einem 3D-Terrain	302
15.1	Führen Sie das Spiel aus	302
15.2	Bereiten Sie das Spiel vor	305
15.2.1	Erzeugen Sie Projekt und Landschaft	305
15.2.2	Steuern Sie den Zufall	305
15.2.3	Erzeugen Sie die weiteren Spielobjekte	309
15.2.4	Erstellen Sie die drei Prefabs	310
15.2.5	Zoomen Sie mithilfe eines Sliders	311
15.3	Erstellen Sie den Spielablauf	313
15.3.1	Bewegen Sie den Jäger	314
15.3.2	Treffen Sie die Ziele	316
15.3.3	Die Ziele starten eine Abwehr	318
15.3.4	Die Abwehr wird gefährlich	320
15.3.5	Messen Sie die Zeit	322
15.3.6	Ideen für Ihre Erweiterungen	324
16	Eine Schlange aus 3D-Joints	325
16.1	Führen Sie das Spiel aus	325
16.2	Bereiten Sie das Spiel vor	326
16.2.1	Erzeugen Sie die Assets und die Platte	326

16.2.2	Erstellen Sie die Schlange und ihre Beute	327
16.2.3	Stellen Sie die gelenkigen Verbindungen her	329
16.3	Erstellen Sie den Spielablauf	330
16.3.1	Bewegen Sie die Schlange	330
16.3.2	Treffen Sie die Beute	331
16.3.3	Verkürzen Sie die Schlange	332
16.3.4	Zählen Sie die Punkte	334
16.3.5	Die Segmente treffen den Rand	335
16.3.6	Messen Sie die Zeit	336
16.3.7	Ideen für Ihre Erweiterungen	338
17	Ein Renntraining und ein Autorennen	339
17.1	Führen Sie das Renntraining aus	339
17.2	Führen Sie das Autorennen aus	341
17.3	Bereiten Sie das Renntraining vor	342
17.3.1	Erzeugen Sie das Projekt und die Fahrbahn	343
17.3.2	Konstruieren Sie das Fahrzeug	344
17.3.3	Fügen Sie die Wheel Collider hinzu	345
17.4	Erstellen Sie den Ablauf des Renntrainings	347
17.4.1	Beschleunigen Sie das Fahrzeug	347
17.4.2	Lenken Sie das Fahrzeug	348
17.4.3	Folgen Sie dem Fahrzeug mit der Kamera	350
17.4.4	Bauen Sie die Begrenzungen auf	352
17.4.5	Eine »Lichtschranke« an der Startlinie	354
17.4.6	Messen Sie die Rundenzeiten	356
17.5	Erweitern Sie das Renntraining zum Autorennen	358
17.5.1	Erzeugen Sie das zweite Fahrzeug	359
17.5.2	Steuern Sie die Fahrzeuge getrennt	360
17.5.3	Teilen Sie den Bildschirm auf	362
17.5.4	Eine dritte Kamera für den Überblick	364
17.5.5	Getrennte Rundenzeiten nach einem Countdown	365
17.5.6	Ideen für Ihre Erweiterungen	369

18

Erkunden Sie das Verlies

370

18.1

Führen Sie das Spiel aus

370

18.2

Bereiten Sie das Spiel vor

375

18.2.1

Die Planung des Verlieses

376

18.2.2

Der Aufbau einer Kammer

376

18.2.3

Erstellen Sie die ersten Spielobjekte

378

18.2.4

Bauen Sie das Prefab für die Kammer

379

18.2.5

Die Schlüssel, Kisten und Sperren

381

18.2.6

Gestalten Sie die Benutzeroberfläche

383

18.3

Erstellen Sie den Spielablauf

384

18.3.1

Folgen Sie dem Spieler mit der Kamera

385

18.3.2

Erstellen Sie alle Kammern

386

18.3.3

Konfigurieren Sie die Kammern

388

18.3.4

Gehen Sie durch ein Tor

391

18.3.5

Nehmen Sie den Schlüssel aus einer Schatzkiste

394

18.3.6

Schließen Sie eine Sperre auf

398

18.3.7

Speichern Sie den Spielstand

400

18.3.8

Laden Sie den alten Spielstand

401

18.3.9

Ideen für Ihre Erweiterungen

402

19

Ein Programmierkurs in C#

404

19.1

Das Unity-Projekt »Programmierkurs«

404

19.2

Grundlagen

406

19.2.1

Variablen und Datentypen

406

19.2.2

Rechenoperatoren

408

19.2.3

Division von ganzen Zahlen

411

19.2.4

Verzweigungen

411

19.2.5

Logische Verknüpfungen

413

19.2.6

Schleifen und Zufallszahlen

415

19.3

Datenfelder

419

19.4

Zeichenketten

423

19.5

Methoden

425

19.5.1

Einfache Methode

426

19.5.2

Methode mit Parametern

426

19.5.3

Methode mit Rückgabewert

427

19.5.4

Methode mit Verweis-Parameter

429

19.6

Generische Listen

430

19.6.1

Hilfsmethode »AusgabeListe()«

432

19.6.2

foreach-Schleife

433

19.7

Daten auf der Festplatte

434

19.7.1

Daten speichern

434

19.7.2

Daten laden

435

19.7.3

Kontrolle der Daten

436

19.8

Objektorientierung

437

19.8.1

Die Spielobjekte im »Unity Editor«

438

19.8.2

Die Klasse »Spieler«

440

19.8.3

Änderungen aller Objekte der Klasse

441

19.8.4

Änderungen einzelner Objekte

442

20

Speichern Sie eine Highscore-Liste

444

20.1

Definition der eigenen Klasse

444

20.2

Nutzung der eigenen Klasse

446

20.2.1

Generische Liste erzeugen und füllen

447

20.2.2

Generische Liste anzeigen

448

20.2.3

Einen neuen Eintrag hinzufügen

449

20.2.4

Alles speichern, alles löschen

452

20.2.5

Der Anzeige-Schalter

453

21

Arbeiten Sie mit mehreren Szenen

455

21.1

Der Ablauf des Projekts

455

21.2

Der Aufbau der ersten Szene

456

21.3

Weitere Szenen

460

22

Allgemeine Hinweise

462

22.1

Projekte bearbeiten

462

22.1.1

Projekte umbenennen

462

22.1.2

Projekte kopieren

463

22.2

Unity Packages

464

22.2.1

Exportieren Sie ein Unity Package

464

22.2.2

Importieren Sie ein Unity Package

465

22.3

Unity unter anderen Betriebssystemen

466

22.3.1

Unity unter macOS installieren

466

22.3.2

Unity unter Ubuntu Linux installieren

467

22.4

Browser-Anwendungen erstellen

468

22.5

Android-Apps erstellen

470

22.5.1

Änderungen im Code

470

22.5.2

Installieren Sie den Android Build Support

472

22.5.3

Stellen Sie die Player Settings ein

472

22.5.4

Führen Sie den Android-Build durch

474

22.5.5

Starten Sie die App unter Android

474

22.6

Bonusprojekte

475

22.6.1

Bonusprojekt »TomsFrogger«

475

22.6.2

Bonusprojekt »TomsPacman«

476

Index

478