mitp
Hans-Georg Schumann
Spiele programmieren mit JavaScript FÜR KIDS

Inhalt

Vorwort 13

Einleitung 17

1

Das erste Projekt 21
- In Babylon einsteigen 21
- Ein Objekt positionieren 25
- Die Sache mit Create 29
- Funktionen 31
- Objekte, Vektoren und mehr 36
- Zusammenfassung 39
- Ein paar Fragen 40
- ... und eine Aufgabe 40

2

HTML und Babylon 41
- Get Zip 42
- Abnabelung 1: babylon.js 48
- Abnabelung 2: Notepad++ 51
- Ansehen und starten? 56
- Zusammenfassung 59
- Ein paar Fragen 60
- ... aber keine Aufgabe 60

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

3

Projekt-Erweiterung 61
INDEX.HMTL und GAME.JS 62
Funktionsanweisungen 67
Tasten-Abfrage mit »if« 70
Fehlersuche 76
Grenzkontrollen 78
Zusammenfassung 83
Ein paar Fragen 84
... und ein paar Aufgaben 84

4

Spiele-Physik 85
Objekt-Material 86
Kacheln, Alpha und Color 90
Masse und Gravitation 93
Krafteinwirkung 98
Eigene Funktionen 101
Zusammenfassung 108
Ein paar Fragen 109
... und zwei Aufgaben 109

5

Mit und ohne Grenzen 111
Eine eigene js-Datei 112
Mauerwerk 117
Kameraführung 122
Kugeln schubsen 124
Zufallszahlen und Schleifen 128
Zusammenfassung 130
Ein paar Fragen 131
... und einige Aufgaben 131

6

Kleiner Krabbelkurs 133
Neues Spiel? 133
Sprites 136
Insekt als Player 139
Die Sache mit der Maus 144
Klick-Wanderung 147
Zusammenfassung 152

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Ein paar Fragen ... 153
... und eine Aufgabe 153

Wanzenjagd 155
Freilauf 155
Klick und Platt 161
Zielen und treffen 165
Eine Horde von Wanzen 168
Zusammenfassung 174
... und zwei Aufgaben 175

Sightseeing 177
Ein leeres Spielfeld 177
Gravitation und Kollision 181
Ein neuer Player 186
Spielfeld mit Hindernissen 189
Kontaktaufnahme 193
Zusammenfassung 197
Ein paar Fragen ... 198
... und eine Aufgabe 198

Landschaften 199
Ein Terrain 199
Heightmaps 202
Terrain-Texturen 206
Bäume 210
Bumpmaps 214
Zusammenfassung 217
Ein paar Fragen ... 218
... und eine Aufgabe 218

Die vier Elemente 219
Eine Skybox für den Himmel 219
Wasser für den See 225
Baumaterial »herstellen« 230
Die Hütte zusammenbauen 234

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by
mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Zusammenfassung 239
Ein paar Fragen 240
... und ein paar Aufgaben 240

11

Android selbst gemacht 241
Kopf, Rumpf und Glieder 241
Alles zusammen 246
Face Texture 249
Der Android bewegt sich 251
Kollisionsprobleme? 257
Zusammenfassung 260
Ein paar Fragen 261
... und eine Aufgabe 261

12

Animationen 263
Keyframes 263
Arm- und Beingymnastik 268
Walking 271
Start und Stopp 275
Mehr Animationen? 278
Zusammenfassung 281
Ein paar Fragen 282
... und zwei Aufgaben 282

13

Kontakt-Spiele 283
Bewegung im Kugelfeld 283
Kontaktsuche 286
Aus Kugeln werden Bäume 290
2D in 3D 294
Treffer sehen und hören 298
Spiel auf Zeit 301
Zusammenfassung 304
Ein paar Fragen 305
... und ein paar Aufgaben 305
Zum Schluss 305

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by
mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Anhang **307**
Für Eltern 307
... und für Lehrer 308

Anhang **311**
Der Babylon-Playground 311

Anhang **317**
Browser-Vielfalt 317

Anhang **323**
Kleine Checkliste 323
Kleine Hilfsmittel 324

Stichwortverzeichnis **325**

A

B

C

D

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by
mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Vorwort

Eigene Fantasiewelten erschaffen, in denen man sich frei bewegen kann. Selbst gebauten Figuren begegnen. Abenteuer selbst erfinden, den Verlauf von Ereignissen selbst bestimmen: Wie wäre das?

Um ein Spiel selbst zu erstellen, muss man vom Programmieren anfangs eigentlich noch gar nichts verstehen. Denn zuallererst braucht man eine Idee und dann einen Plan.

Wovon soll das Spiel handeln? Welche Geschichte soll es erzählen? Personen, Orte und Ereignisse, all das führt zu einem Plan, der umfasst, was zu diesem Spiel gehören soll. Und erst wenn der Plan »steht«, kann die eigentliche Umsetzung in ein Programmprojekt beginnen. Dann allerdings sollte man schon möglichst gut programmieren können.

Wir wollen hier gar nicht so hoch hinaus: Ein professionelles Game wird heutzutage ja von einer ganzen Gruppe von Leuten erstellt, darunter Designer, Künstler, Techniker und nicht zuletzt natürlich Programmierer.

Trotzdem dauert die Arbeitszeit häufig mindestens Monate, wenn nicht Jahre. Die Beteiligten machen einen Vollzeitjob, es ist ihr Beruf. Hier hast du als Einzelgänger nur eine Chance, wenn deine Spielidee so hervorragend und einmalig ist, dass sie alles andere überstrahlt.

Bleiben wir also auf dem harten Boden der Tatsachen und planen wir nicht ein gigantisches Meisterwerk, sondern kümmern wir uns um solide Grundlagen. Wenn du die beherrschst, hast du durchaus Voraussetzungen, auch einmal an einem professionellen Spielprojekt mitzuwirken.

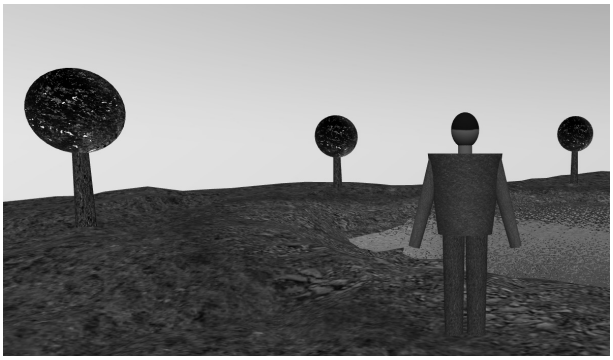
© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by
mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Welche Werkzeuge benötigen wir?

Um Spiele im 2D- und 3D-Bereich zu erstellen, brauchen wir als Herzstück eine sogenannte **Engine**. Sie muss mit physikalischen Gesetzen umgehen können, damit die Spielwelt mit ihren Figuren und Ereignissen möglichst echt wirkt. Und sie muss grafische Effekte beherrschen, damit das Ganze auch optisch etwas hermacht.

Und das brauchst du:

- die **Babylon Engine**, ein vollwertiges System, das vielfältige Möglichkeiten zum Erstellen von auch professionellen Spielen bietet.



- einen Editor, der dich beim Programmieren unterstützt. Mehr als Notepad, den Windows bereits »eingebaut« hat, bietet **Notepad++**, das wir in diesem Buch einsetzen.

Programmiert wird in der Sprache **JavaScript**, in der auch die komplette Babylon Game Engine erstellt wurde.

- Damit die Programme funktionieren, benötigen wir also einen **Browser**, wie du ihn für das Surfen im Internet benutzt: Ob Google Chrome, Microsoft Edge oder Mozilla Firefox, jeder Browser versteht JavaScript. (Näheres über einige Unterschiede erfährst du im Anhang.)

Wir verwenden hier die derzeit aktuelle Babylon-Version 2.5. Die bekommst du zum Herunterladen auf *http://www.babylonjs.com*. Dort erfährst du auch, ob es eine neuere Version gibt. Alle Programme im Buch funktionieren mit BABYLON.JS 2.5. Bei deutlich höheren Versionsnummern müssen vielleicht einige Programme neu angepasst werden.

Einen komfortablen Editor zum Bearbeiten deiner Programme findest du auf *https://notepad-plus-plus.org*.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Und was bietet dieses Buch?

Vorwiegend geht es um die Programmiersprache JavaScript und natürlich um Spiele. Du erfährst hier unter anderem

- einiges über JavaScript
- etwas über Tasten- und Maussteuerung
- wie man 2D-Spiele mit Sprites erstellt
- wie man Material und Texturen einsetzt
- wie man eine Landschaft zum Spielfeld macht
- wie man einen Androiden selber baut
- etwas über den Umgang mit Kollisionen
- wie man Objekte animiert

Im **Anhang** gibt es dann noch zusätzliche Informationen, unter anderem über Anpassungsprobleme und den Umgang mit Fehlern.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Einleitung

Wie arbeite ich mit diesem Buch?

Um dir den Weg vom ersten Projekt bis zu einem 3D-Game einfacher zu machen, gibt es einige zusätzliche Symbole, die ich dir hier gern erklären möchte:

Arbeitsschritte

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

→ Grundsätzlich lernt man besser, wenn man etwas selber programmiert. Aber nicht immer hat man große Lust dazu. Weil es alle Projekte im Buch auch zum Download gibt, findest du am Ende des einen oder anderen Abschnitts auch den jeweiligen Dateinamen (zum Beispiel → MYGAME1). Wenn du also das Projekt nicht selbst erstellen willst, kannst du es stattdessen auch aus dem Internet laden – zu finden unter *www.mitp.de/577*

Fragen und Aufgaben

Am Ende eines Kapitels gibt es jeweils eine Reihe von Fragen und Aufgaben. Diese Übungen sind nicht immer ganz einfach, aber sie helfen dir, deine Spiele noch besser zu entwickeln. Lösungen zu den Aufgaben findest du ebenfalls auf der **mitp**-Homepage. Du kannst sie dir alle im Editor von Windows oder auch in deinem Textverarbeitungsprogramm anschauen. Oder du lässt sie dir ausdrucken und hast sie dann schwarz auf weiß, um sie neben deinen PC zu legen.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Notfälle

Vielleicht hast du irgendetwas falsch gemacht oder etwas vergessen. Oder es wird gerade knifflig. Dann fragst du dich, was du nun tun sollst. Bei diesem Symbol findest du eine Lösungsmöglichkeit. Auch ganz hinten im Anhang D findest du ein paar Hinweise zur Pannenhilfe.

Wichtige Stellen im Buch

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

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

Was brauchst du für dieses Buch?

Du findest die Babylon Engine als komplette Entwicklungsumgebung zum Download auf dieser Homepage:

http://www.babylonjs.com

Babylon ist kostenlos, du kannst dir dort sogar dein System zusammenstellen, indem du es mit verschiedenen Zusätzen erweiterst.

Die Beispielprojekte in diesem Buch findest du ebenso wie die Lösungen zu den Aufgaben auf der Homepage des Verlages in der gerade aktuellen Version:

http://www.mitp.de/577

Alle Projekt-Ordner enthalten nur die Beispiel-Programme, **nicht** aber die Babylon Engine – aus Copyright-Gründen. Du musst also selbst die Babylon-Datei herunterladen und in den jeweiligen Projekt-Ordner kopieren.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Betriebssystem

Die meisten Computer arbeiten heute mit dem Betriebssystem Windows. Davon brauchst du eine der Versionen ab 7. Mit Babylon lassen sich übrigens nicht nur Spiele für Windows entwickeln, sondern auch für andere Systeme.

Speichermedien

Auf jeden Fall benötigst du etwas wie einen USB-Stick oder eine SD-Card, auch wenn du deine Programme auf die Festplatte speichern willst. Auf einem externen Speicher sind deine Arbeiten auf jeden Fall zusätzlich sicher aufgehoben.

Gegebenenfalls bitte deine Eltern oder Lehrer um Hilfe: Sie sollten den Anhang A lesen. Dann können sie dir bei den ersten Schritten besser helfen.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

1

Das erste Projekt

Du möchtest natürlich gleich dein erstes Spiel erstellen. So schnell geht das leider nicht, aber mit dem Programmieren können wir sofort loslegen. Du brauchst dazu anfangs nur den Browser, mit dem du auch im Internet surfst. Dort benutzen wir eine »Spiel-Maschine« namens Babylon, mit der lassen sich jede Menge Spiele erstellen. Aber wir fangen erst mal mit etwas Einfachem an.

In diesem Kapitel lernst du

- wie man in Babylon einsteigt
- wie man Objekte in einem Projekt einsetzt
- etwas über 2D und 3D
- wie man die Position von Objekten ändert
- was Funktionen und Vektoren sind

In Babylon einsteigen

Starte einen Browser deiner Wahl, das kann zum Beispiel Microsoft Edge, Google Chrome oder Mozilla Firefox sein. Dort gib in der Adresszeile diese Adresse ein:

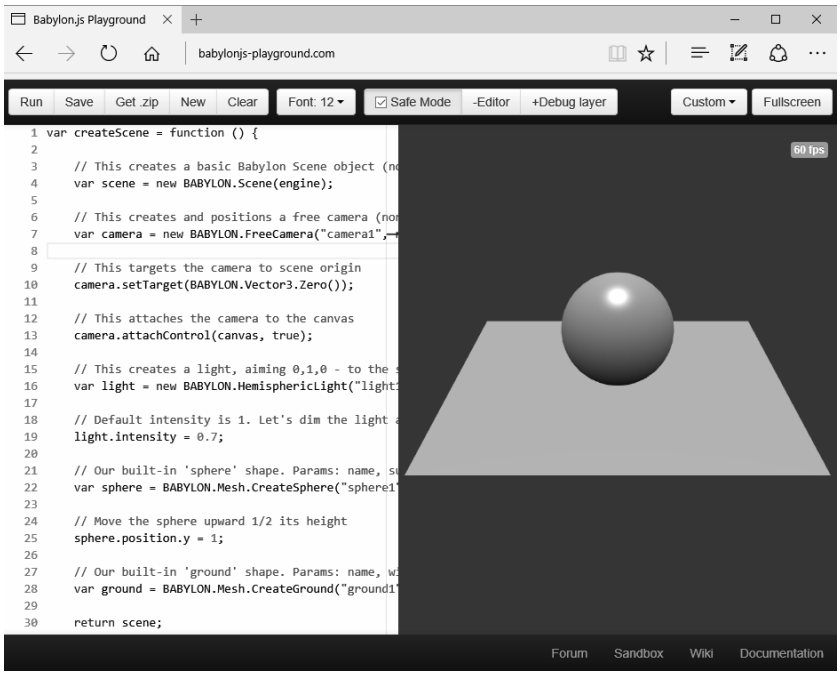
http://www.babylonjs.com/

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Nach einer Weile landest du auf der Homepage von **Babylon JS**. Dort findest du eins der Werkzeuge, die wir für dieses Buch brauchen: ein komplettes System zum Erstellen von Spielen.

Probieren wir das gleich aus.

≫ Klicke auf die Schaltfläche TRY, um in den sogenannten **Playground** umzuschalten.



© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Auf der rechten Seite siehst du eine Fläche und darüber eine Kugel. In diesem Bereich findet auch später dein Spiel statt.

Man nennt das Ganze auch **Spielwelt**, denn natürlich lässt sich dieser Bereich später auf den ganzen Bildschirm vergrößern. Wie bei einem echten Spiel kann man dann auch nach rechts oder links weiterwandern. Das alles gehört zur Spielwelt (die mitunter sogar riesig sein kann).

Die Fläche unter der Kugel ist so etwas wie das **Spielfeld**. Und die Kugel selbst ist dann die **Spielfigur**. Und damit kennst du nun das, was mindestens zu einem Spiel gehört: eine Spielfigur und ein Spielfeld.

Auf der linken Seite deines Browsers befindet sich furchtbar viel Text. Alles, was da steht, ist in der Programmiersprache **JavaScript** geschrieben. Und das ist auch die Sprache, in der du hier vorwiegend programmieren wirst. (Und jetzt kannst du dir wohl denken, was das **»JS«** hinter »Babylon« bedeutet, oder?)

Es bleibt dir nicht erspart, sich näher mit dem »ganzen Kram« zu befassen, der da steht. Ein besserer Ausdruck dafür ist **Quelltext**. Wir gehen das Ganze gleich langsam Zeile für Zeile durch. Genau genommen ist es gar nicht so viel, wie es zuerst aussieht.

Ich habe alle Zeilen, die mit einem doppelten Schrägstrich beginnen, einmal entfernt, weil sie für das Funktionieren des Programms nicht nötig sind. Es handelt sich um sogenannte **Kommentare**, in denen (auf Englisch) erläutert wird, was an der jeweiligen Stelle passiert (oder passieren soll). Solche Zeilen werden von Babylon einfach übersprungen.

Was übrig bleibt, sieht so aus:

```
var createScene = function () {
    var scene = new BABYLON.Scene(engine);
    var camera = new BABYLON.FreeCamera
    ("camera1", new BABYLON.Vector3(0,5,-10), scene);
    camera.setTarget(BABYLON.Vector3.Zero());
    camera.attachControl(canvas, true);
    var light = new BABYLON.HemisphericLight
    ("light1", new BABYLON.Vector3(0,1,0), scene);
    light.intensity = 0.7;
    var sphere = BABYLON.Mesh.CreateSphere
    ("sphere1", 16, 2, scene);
```


© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

```
    sphere.position.y = 1;
    var ground = BABYLON.Mesh.CreateGround
    ("ground1", 6, 6, 2, scene);
    return scene;
};
```

In Wirklichkeit hat der Quelltext noch weniger Zeilen, ich musste nur einige sehr lange Exemplare auf zwei Buch-Zeilen verteilen.

Dieser Quelltext ist eine Ansammlung von **Anweisungen**, die der Computer ausführen soll. Jede davon wird mit einem Semikolon (`;`) abgeschlossen. Alle Anweisungen zusammen bewirken das, was du rechts neben dem Text zu sehen bekommst.

Da taucht ständig dieses Wörtchen `var` auf. Das ist eine Abkürzung für **Variable**. Direkt dahinter steht ein Name und dann folgt ein sogenanntes **Zuweisungszeichen** (`=`), das wie ein Gleichheitszeichen aussieht.

var Name = ;

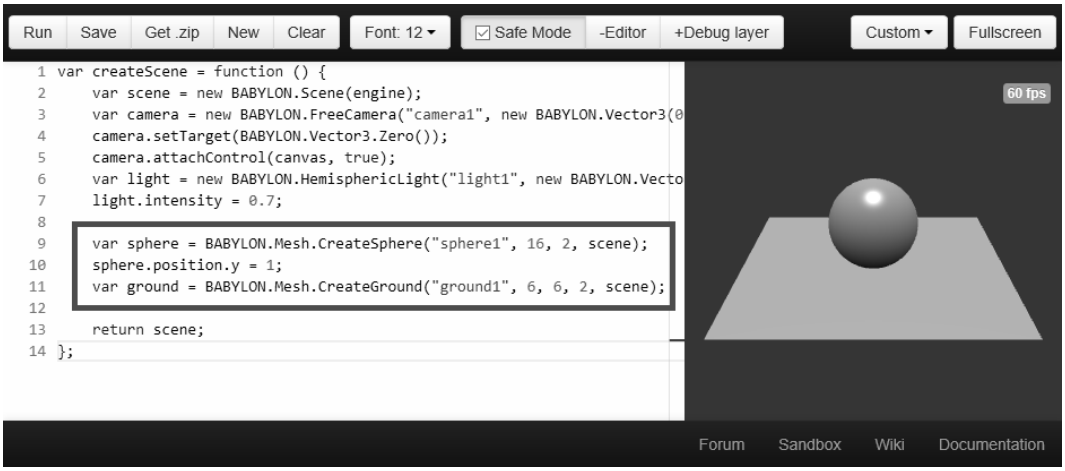
Jedes Mal, wenn `var` erscheint, heißt das: Es gibt etwas Neues, das Programm oder Spiel wird um ein neues Element erweitert. Dabei ist die Variable ein Platzhalter oder eine Art Behälter, dessen Inhalt veränderbar ist.

Und die **Zuweisung** bedeutet, dass in den Behälter etwas »eingefüllt« wird. Hinter dem Zuweisungszeichen kann alles Mögliche stehen. Und wie du oben siehst, werden dem Spiel mithilfe von `var` Elemente **hinzugefügt** (wie Kamera, Licht, Kugel und Untergrund).

Zu beachten ist, dass bei einigen `var`-Zuweisungen ein zusätzliches `new` nötig ist. Ich komme weiter unten darauf zurück.

Schauen wir jetzt mal noch genauer hin. Dabei beginnen wir nicht ganz oben. Das wäre noch zu kompliziert, dazu kommen wir also erst später. Beginnen wir mit den Zeilen, in denen wir selbst schon mal etwas »machen« können.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577



Ein Objekt positionieren

In den beiden Anweisungen, die ich meine, geht es zuerst um ein Objekt namens `sphere`, zu Deutsch: Kugel. Diese Zeilenfolge sorgt dafür, dass nebenan im Fenster eine solche Kugel zu sehen ist:

```
var sphere = BABYLON.Mesh.CreateSphere
("sphere1", 16, 2, scene);
sphere.position.y = 1;
```

Mit `CreateSphere()` wird eine Kugel erzeugt. Und mit `position` wird diese Kugel auf der Unterlage positioniert und nicht »dazwischen«.

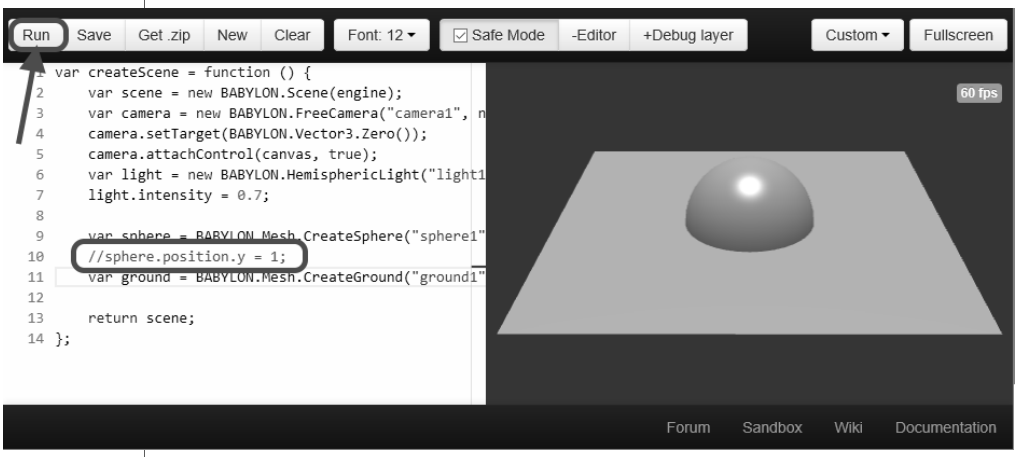
≫ Damit du verstehst, was ich damit meine, ändere die betreffende Zeile einmal so um:

```
// sphere.position.y = 1;
```

Durch die zwei vorgeschalteten Schrägstriche (`//`) wird das Ganze zu einem Kommentar. Es ist damit keine Anweisung mehr und wird nach dem Programmstart übersprungen.

≫ Klicke nun oben links auf RUN, um das Programm neu zu starten.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

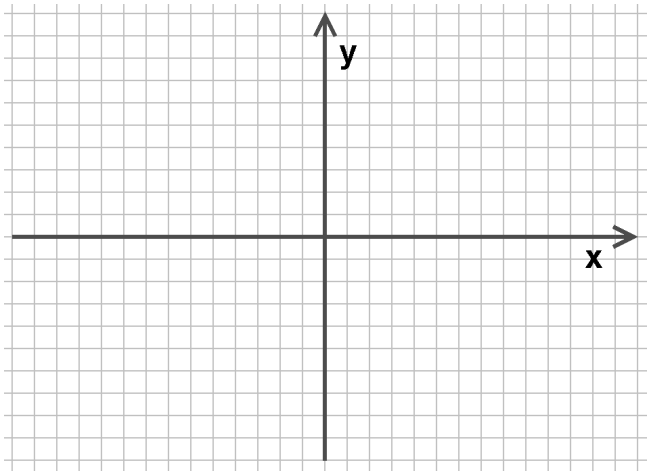


Und schon sackt die Kugel in den Untergrund. Mit dem Zusatz `position` können wir also die Position eines Objekts ändern.

Standardmäßig wird ein neues Objekt erst mal in die **Mitte** einer Spielwelt gesetzt. Das entspricht dann diesem Positionswert:

```
sphere.position.y = 0;
```

Wenn man also für `position` gar nichts angibt, dann wird der Wert auf 0 gesetzt. Aber schauen wir mal genauer hin. Da steht ja auch noch ein `y`. Vielleicht weißt du aus der Schule, was ein **Koordinatensystem** ist. Da gibt es eine x-Achse (= Horizontale oder Waagerechte) und eine y-Achse (= Vertikale oder Senkrechte).

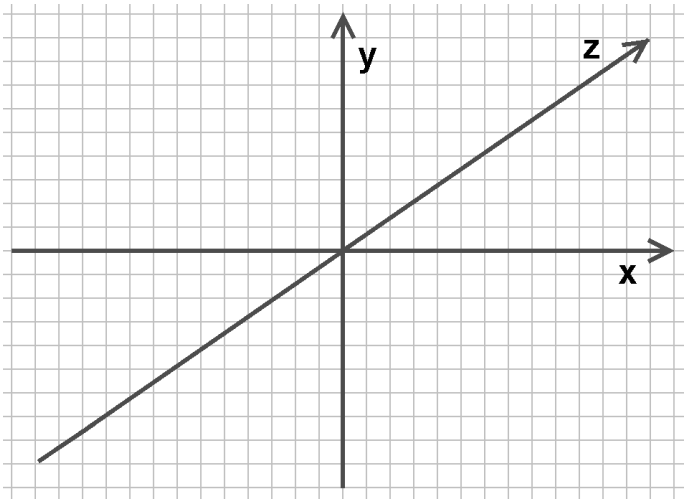


Entlang der x-Achse geht es also nach links oder rechts, entlang der y-Achse nach oben oder unten. Genau in der Mitte, wo sich die Achsen kreuzen, sind die Koordinatenwerte (0 | 0). Das nennt man 2D.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

In Babylon gibt es noch eine dritte Ache, die man z-Achse nennt. An der entlang geht es nach vorn oder nach hinten. Damit haben wir 3D.

Schaut man von vorn auf das Koordinatensystem, dann kann man diese Achse nicht sehen. Um alle Achsen dennoch in 2D sichtbar zu machen, greift man zu einem optischen Trick: Die z-Achse wird als Diagonale dargestellt.



Und so kann man die Richtungen für die Positionierung von Objekten wie zum Beispiel einer Kugel in einer Tabelle zusammenfassen:

Achse	X	Y	Z
Richtung	links-rechts	oben-unten	vorn-hinten

2D oder 3D? Was genau bedeutet das und was ist der Unterschied? Ein Bild auf einem Blatt Papier ist zweidimensional (abgekürzt: 2D), es hat zwei Dimensionen, eine Länge und eine Breite. Dein Computer oder dein Smartphone sind dreidimensional (abgekürzt: 3D), denn dort gibt es zusätzlich noch eine Höhe oder Dicke.

Bei Spielen gibt es das eigentlich nicht, denn die Darstellung auf dem Monitor oder Display ist immer 2D. Die dritte Dimension wird künstlich erzeugt, es sieht dann für die Augen so aus, als wäre das Ganze 3D. Mit Babylon können die Spiele 2D oder 3D sein.

Probieren wir doch gleich mal aus, wie sich die Lage der Kugel verändern lässt.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

- Entferne zuerst die Kommentarzeichen (//). Dann ändere die vorhandene Zeile so um:

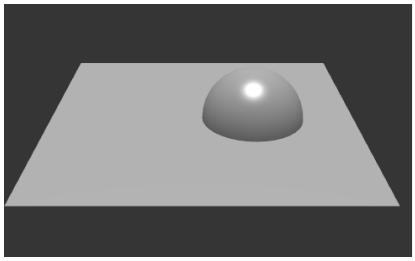
```
sphere.position.x = 1;
```

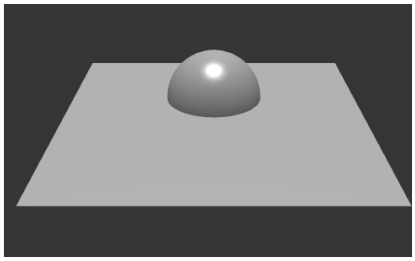
Und dann so:

```
sphere.position.z = 1;
```

- Vergiss nicht, jedes Mal auf RUN zu klicken.

Im einen Fall verschiebt sich die (eingesackte) Kugel nach rechts, im anderen nach hinten:

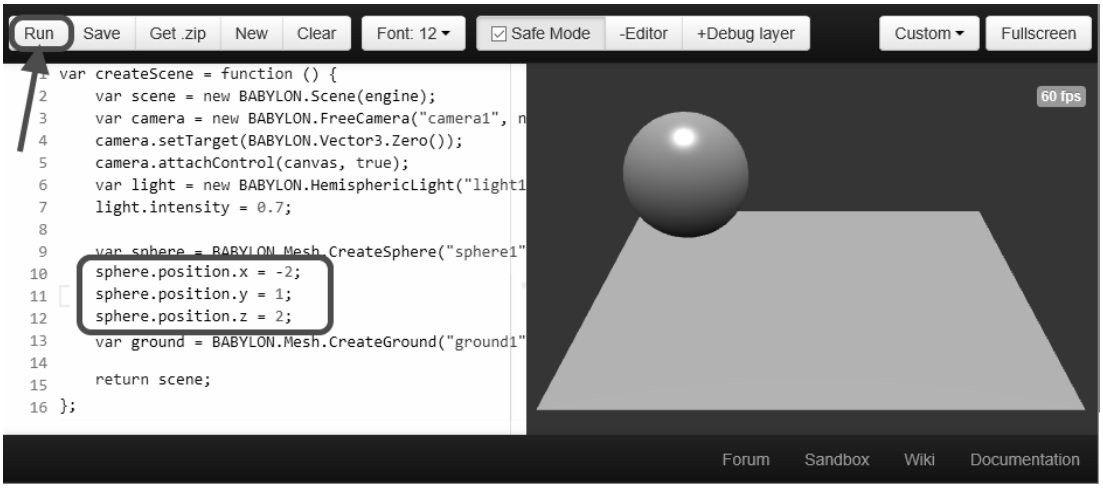




- Nun mache aus der vorhandenen Zeile die folgenden drei:

```
sphere.position.x = -2;
sphere.position.y = 1;
sphere.position.z = 2;
```

- Dann klicke auf RUN.



© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Und du kannst sehen, dass die Kugel nun hinten links in der Ecke liegt, und nicht in, sondern auf der Spielfläche.

Wenn du etwas falsch eingetippt oder versehentlich etwas gelöscht hast, kannst du das in der Regel mit Strg + Z wieder rückgängig machen.

Die Sache mit Create

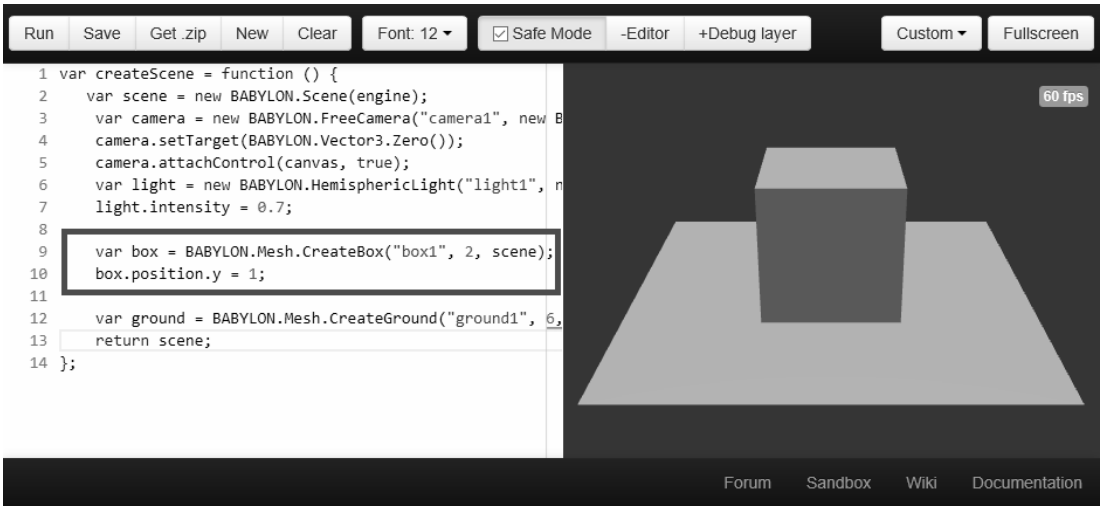
Du weißt also jetzt, wie man eine Kugel als Spielelement erzeugt und in Position bringt. (Spielfigur ist hier noch etwas übertrieben, solange sich das »Ding« nicht bewegen kann. Aber warte ab, das kommt noch.)

Was ist mit anderen Objekten? Da muss es doch noch mehr geben.

≫ Probiere doch mal das Folgende aus: Entferne die Zeilen für die Kugel und ersetze sie durch diese:

```
var box = BABYLON.Mesh.CreateBox("box1", 2, scene);
box.position.y = 1;
```

Damit bekommst du nun einen Würfel.



Weitere Objekte wirst du im Laufe dieses Buches kennenlernen. Nun kümmern wir uns um das Drumherum. Das ist auch wichtig dafür, dass

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

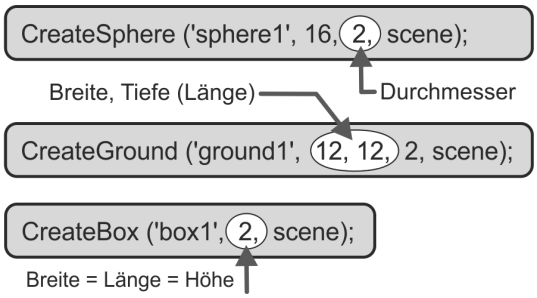
wir eine Kugel oder einen Würfel auf einem Spielfeld zu sehen bekommen. Das Spielfeld, auch Untergrund genannt, wird übrigens so erzeugt:

```
var ground = BABYLON.Mesh.CreateGround
 ("ground1", 6, 6, 2, scene);
```

Das erledigt die Funktion `CreateGround()`. Womit wir nun schon mal diese drei Funktionen kennen:

`CreateSphere()`	Eine Kugel erzeugen
`CreateGround()`	Eine Fläche (als Untergrund) erzeugen
`CreateBox()`	Einen Quader oder Würfel erzeugen

Alle werden mit »Create« eingeleitet, dem englischen Wort für »Erzeugen«. Schon ganz zu Anfang kannst du bestimmen, wie groß das jeweilige Objekt sein soll. Dafür sind sogenannte Parameter da, die in Klammern hinter dem `Create`-Namen stehen.



Eine Zwischenbemerkung zu den **Anführungszeichen** ("), die hier immer wieder auftauchen: Bis jetzt sind dir wohl immer die doppelten begegnet. Wenn du dir Quelltexte im Internet ansiehst, wirst du aber auch oft auf die einfachen Anführungszeichen (') stoßen. In JavaScript sind beide **gleichwertig** erlaubt.

Falls du Probleme hast, die Anführungszeichen auf der Tastatur zu finden, versuch's mal mit ⇧+2 und ⇧+#.

≫ Probiere jetzt diese Anweisungen für das Spielfeld gleich einmal aus:

```
var ground = BABYLON.Mesh.CreateGround
 ("ground1", 15, 15, 2, scene);
```


© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

oder:

```
var ground = BABYLON.Mesh.CreateGround
 ("ground1", 5, 25, 2, scene);
```

Und hier ein Beispiel für eine große Kugel:

```
var sphere = BABYLON.Mesh.CreateSphere
("sphere1", 16, 10, scene);
sphere.position.y = 5;
```

Wie wäre es mit einem kleinen Würfel?

```
var box = BABYLON.Mesh.CreateBox("box1", 0.4, scene);
box.position.y = 0.2;
```

≫ Experimentiere selber mit eigenen Werten.

Wie du siehst, kannst du auch Dezimalzahlen einsetzen. Denk aber daran, dass du hier **kein** Komma, sondern einen **Punkt** verwendest:

```
box.position.y = 0.2; // richtig
box.position.y = 0,2; // falsch
```

Funktionen

Kommen wir jetzt zur ersten Zeile des Quelltextes, wie er seit dem Start im linken Browser-Bereich steht:

```
var createScene = function () {
```

Hier wird eine Funktion definiert, die `createScene` heißt.

Was ist eigentliche eine **Funktion**. Eine Funktion ist eine Methode, eine Arbeitsanweisung für den Computer.

Kurz: Was in einer Funktion steht, führt der Computer aus. Zum Beispiel eine Kugel erzeugen und auf dem Bildschirm beziehungsweise in einem Browser darstellen.

Das Besondere ist, dass du nicht nur Funktionen benutzen kannst, die es bereits gibt – wie zum Beispiel `CreateSphere` –, sondern du kannst dir auch deine eigenen Funktionen programmieren. Und wenn du die dem Computer »befiehlst«, dann führt er sie auch aus.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Funktions-Variablen können in JavaScript diese Form haben:

```
var Name = function () {   };
```

Zuerst kommt das Wort `var`, wie du weißt eine Abkürzung für **Variable**. Dahinter folgt sofort ein **Name**, meistens ein englisches Wort, man könnte aber auch ein deutsches nehmen, zum Beispiel:

```
var laufLos = function () {   };
var gibMirGeld = function () {   };
```

Wenn ein Name aus mehreren Teilen besteht, kann man ihn nicht durch ein Leerzeichen oder Bindestriche trennen. Dann setzt man mittendrin einzelne Großbuchstaben ein. Ein Name muss immer eine **zusammenhängende** Zeichenkette sein.

In JavaScript wird zwischen Groß- und Kleinschreibung unterschieden: `laufLos` und `lauflos` und `Lauflos` sind also nicht dasselbe!

Warum es einerseits `createScene()` und andererseits `CreateSphere()` heißt? Vielleicht, weil das erste eine selbst definierte Funktion ist, das zweite eine in Babylon bereits eingebaute.

Nach dem Namen folgt ein Zuweisungszeichen (=). Das anschließende Wort `function` besagt, dass es sich um eine Funktion handelt.

Bis hierhin dürfte es klar sein, doch was sollen die vielen Klammern dahinter?

Das erste Paar von runden Klammern ist in unserem Beispiel leer. In JavaScript **müssen** hinter `function` **immer** solche Klammern stehen.

Manche Funktionen übernehmen auch **Parameter**, wie du schon zum Beispiel bei `CreateSphere()` gesehen hast. Parameter sind also Elemente, die eine Funktion übernehmen und intern weiter benutzen kann. Auf Parameter kommen wir später immer wieder zu sprechen.

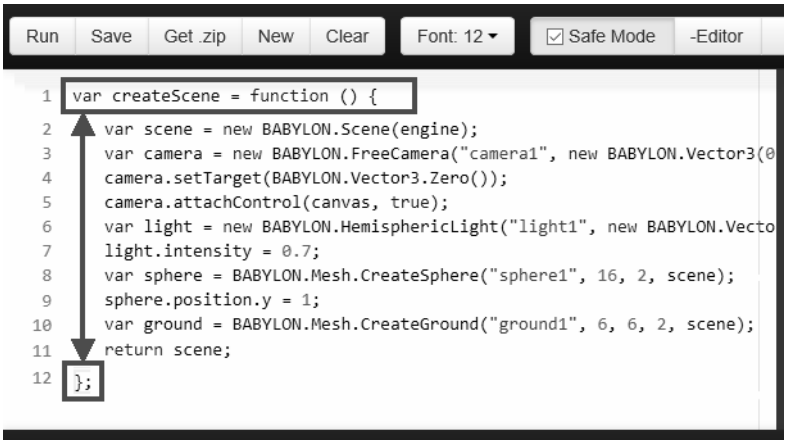
Neben den Funktionen gibt es noch ganz normale Variablen zum Beispiel für Zahlen. Die haben **keine** Klammern. Du wirst auch diesen Typ noch kennenlernen.

Nun zum zweiten Paar, den geschweiften Klammern. Die umfassen das, was eine Funktion ausmacht, nämlich alle Anweisungen.

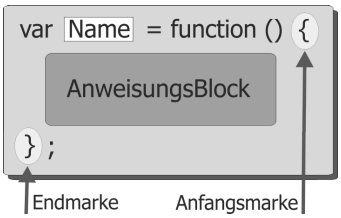
© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Wenn du die Klammern auf deiner Tastatur nicht findest: Für die runden benutzt du ⇧+8 und ⇧+9, für die geschweiften AltGr+7 und AltGr+0.

Nun rutsche mal ein bisschen zurück, damit du den Quelltext komplett im Blick hast.



Wie du siehst, ist das Ganze eine einzige Funktion, die in der ersten Zeile mit der öffnenden geschweiften Klammer endet. Und ganz unten zum Schluss, in der letzten Zeile, da findest du die schließende Klammer.



Beide Klammern sind Marken, sie kennzeichnen den Anfang und das Ende eines Anweisungsblocks. Und dessen **Inneres** schauen wir uns jetzt an.

```
var scene = new BABYLON.Scene(engine);
```

Schon wieder was mit `var`. Diesmal handelt es sich um eine **Objekt**-Variable namens `scene`. Frei übersetzt heißt das: Eine Spielszene in Babylon wird erzeugt. Als Nächstes geht es um die Erzeugung einer Kamera:

```
var camera = new BABYLON.FreeCamera
("camera1", new BABYLON.Vector3(0,5,-10), scene);
```


© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Die Kamera ist nötig, damit du im Spiel überhaupt etwas von der Szene siehst. Sie lässt sich natürlich auch frei bewegen, um durch die Spielszene zu wandern. Zum Start wird die Kamera auf den Mittelpunkt der Spielwelt justiert:

```
camera.setTarget(BABYLON.Vector3.Zero());
```

Das erledigt die Funktion `setTarget`. Und da ist noch eine Zeile, die sich mit der Kamera befasst:

```
camera.attachControl(canvas, true);
```

Die Funktion `attachControl()` sorgt dafür, dass die Kamera steuerbar wird. Das merkst du daran, dass du mit der Maus und den Tasten etwas bewegen kannst.

- Probiere das mal aus: Klicke in den rechten Bereich mit der Kugel und der Unterlage. Bewege die Maus mit gedrückter Maustaste. Dann drücke eine der Pfeiltasten.
- Anschließend setze Kommentarzeichen (`//`) vor die `attachControl`-Zeile und wiederhole das Ganze. Diesmal dürfte das Verschieben der Szene nicht mehr klappen. Vergiss nicht, die Schrägstriche wieder zu entfernen.

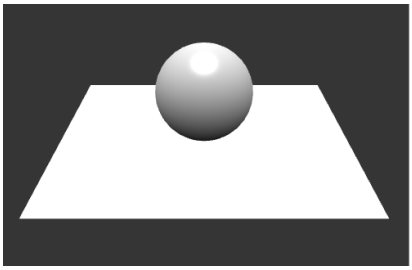
Neben der Kamera ist natürlich das Licht wichtig, das für die Sichtbarkeit einer Szene zuständig ist:

```
var light = new BABYLON.HemisphericLight
("light1", new BABYLON.Vector3(0,1,0), scene);
```

Nach dem »Anschalten« wird noch die Lichtstärke etwas heruntergeregelt (voreingestellt ist der Wert 1):

```
light.intensity = 0.7;
```

Du kannst ja mal mit verschieden Werten experimentieren. So sieht es bei mir aus, wenn die Lichtintensität auf 1 eingestellt ist (links) oder wenn das Licht aus ist (rechts):



© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

≫ Es gibt natürlich auch die Möglichkeit, dem Licht höhere Werte als 1 zu geben. Schau mal, wie es aussieht, wenn du `light.intensity` auf 3 oder mehr setzt.

Nun kommen einige Zeilen, die du schon kennst. Hier werden Spielfigur und Spielfeld erzeugt, auch das läuft über Objekt-Variablen:

```
var sphere = BABYLON.Mesh.CreateSphere
("sphere1", 16,2, scene);
sphere.position.y = 1;
var ground = BABYLON.Mesh.CreateGround
("ground1", 6,6,2, scene);
```

Ganz zum Schluss siehst du noch eine Zeile, die ziemlich unscheinbar aussieht:

```
return scene;
```

Mit der Anweisung `return` ist die Funktion `createScene` zu Ende. (Auf die genaue Bedeutung dieser letzten Zeile kommen wir später noch zurück.)

Nicht zu vergessen ist die abschließende geschweifte Klammer. (Das Semikolon ist an dieser Stelle nicht zwingend nötig.)

Hier sind noch mal alle Schritte zusammen, die mindestens für eine Spielszene nötig sind – sozusagen als »Kochrezept«:

1. Spielszene erzeugen

```
var scene = new BABYLON.Scene(engine);
```

2. Kamera "aufstellen"

```
var camera = new BABYLON.FreeCamera("camera1",
new BABYLON.Vector3(0,5,-10), scene);
camera.setTarget(BABYLON.Vector3.Zero());
camera.attachControl(canvas, true);
```

3. Licht "einschalten"

```
var light = new BABYLON.HemisphericLight("light1",
new BABYLON.Vector3(0,1,0), scene);
```

4. Objekte erzeugen

```
var object = BABYLON.Mesh.CreateXXX(xxx, scene);
```

In der letzten Anweisung steht das »XXX« für die jeweilige Figur, die man verwenden will (zum Beispiel Kugel, Quader, Person, Fahrzeug, Fläche).

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Davon hängt dann auch ab, welche Parameter dahinter gebraucht werden – deshalb ein weiteres »xxx«.

Objekte, Vektoren und mehr

Da ist einiges an Objekt-Variablen zusammengekommen. Ich fasse sie alle einmal in einer Tabelle zusammen:

scene	Die Spielszene, auch Level genannt
camera	Die Kamera, um an jede Stelle der Spielwelt zu gelangen
light	Das Licht, das die gesamte Szene erleuchtet
sphere	Die Spielfigur, hier nur eine Kugel
ground	Das Spielfeld, hier ein einfacher Untergrund

Und damit hast du auch die Grundzutaten für fast jedes Spiel. In einem größeren Spiel gibt es unzählige Objekte, darunter natürlich auch viele, die sich bewegen (lassen).

An dieser Stelle komme ich noch mal auf das `new` zurück. Wird einer Variablen eine Funktion (wie zum Beispiel `CreateSphere()`) zugewiesen, so geht das direkt, die Variable erhält das, was die Funktion **zurückgibt**.

```
var Name = Funktionsname();
```

```
var Name = new Objektname();
```

Wird einer Variablen ein Objekttyp (wie zum Beispiel `FreeCamera`) zugewiesen, muss das Objekt **neu** erzeugt werden. Daher wird hier ein `new` vorangestellt.

Hier bleiben wir erst einmal noch bei den Zutaten, die Babylon uns beim ersten Start anbietet. Doch schon im nächsten Kapitel basteln wir an unserer Spielwelt herum.

Natürlich reichen meine Erläuterungen nicht aus, um jedes einzelne Wort im Quelltext zu verstehen. Was mit dem großgeschriebenen `BABYLON`

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

gemeint ist, dürfte klar sein, damit sprechen wir direkt die »Spiel-Maschine« (englisch Game Engine) an, mit der wir es hier zu tun haben.

Was hat es mit dem Wort `Vector3` auf sich, das immer wieder auftaucht?

Der Begriff **Vektor** stammt aus der Mathematik. Mit einem Vektor lässt sich ein Punkt in einem Koordinatensystem festlegen. Den Mittelpunkt unserer Spielwelt bezeichnen wir mit dem Vektor (0, 0, 0). Dabei stehen die einzelnen Nullen für die x-, y- und z-Koordinaten.

Der Name `Vector3` drückt aus, dass wir es hier mit dreidimensionalen Vektoren zu tun haben. Natürlich gibt es in Babylon auch einen `Vector2`.

`Vector.Zero()` übrigens bedeutet, dass **alle** Koordinaten eines Vektors auf 0 gesetzt werden.

Zu anderen Wörtern werde ich hier jetzt nichts weiter sagen, die meisten lernst du nach und nach in den folgenden Kapiteln kennen.

Widmen wir uns jetzt noch mal der Arbeitsumgebung von Babylon im Browser, eher wir diesen wieder verlassen. Bis jetzt haben wir oben in der Menüleiste nur die Schaltfläche RUN benutzt. Doch die anderen Optionen solltest du auch noch kurz kennenlernen.

Hier steht alles Wichtige in einer Tabelle:

RUN	Das links stehende Programm starten und rechts anschauen, was passiert
SAVE	Eine Szene im »Playground« speichern, damit sie später wieder zur Verfügung steht. An die Adresse der Website wird dann eine Zeichenkette angehängt, sodass sie zum Beispiel so aussieht: *http://www.babylonjs-playground.com/#DNCK5* Unter einer solchen Adresse kannst du deinen Quelltext dann erreichen.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

GET.ZIP	Die aktuelle Szene als komprimierte Datei mit dem Namen SAMPLE.ZIP auf deiner Festplatte speichern
NEW	Ein neues Projekt erzeugen
CLEAR	Den ganzen Quelltext löschen
FONT SIZE	Die Schriftgröße des Quelltextes einstellen
SAFE MODE	Sicheren Modus einstellen: Wenn du deinen Quelltext geändert hast, wirst du gewarnt, ehe du die Seite verlässt.
EDITOR	Den Quelltext ausblenden, damit du die Szene im ganzen Browserfenster sehen kannst
DEBUG LAYER	Ein Hilfsmittel dazuschalten, mit dem du Fehlern besser nachspüren kannst
BASIC SCENE	Wechseln auf verschiedene Szenen und Quelltexte, die Babylon zum Anschauen und Ausprobieren anbietet
FULLSCREEN	Vollbildanzeige auf dem Monitor

Ich empfehle dir, außer der hier kennengelernten BASIC SCENE noch einige weitere Szenen durchzutesten, die Babylon dir anbietet. Dabei kannst du auch schon mal einen Blick auf die jeweiligen Quelltexte werfen, wenn du willst.

Mit dem Ausprobieren allerdings solltest du vorsichtig sein. Vor allem mit dem Ändern des Textes, den du da siehst. Vielleicht ist dir nach einem Klick auf RUN eine ähnliche Meldung schon einmal begegnet:

Compilation bedeutet, dass der Computer versucht hat, den Quelltext zu einem ausführbaren Programm zu machen (man könnte auch sagen: zu einer App, die läuft). Und **Error** heißt, dass er dabei auf einen Fehler gestoßen ist, der es ihm unmöglich macht, das Programm auszuführen.

In der Regel steht darunter ein Hinweis, wo der Fehler stecken könnte. Hier ist es eine vergessene Klammer. Oder du hast ein Wort falsch geschrieben? Oft fehlt auch ein anderes Zeichen (Punkt, Komma, Semikolon).

Finde den Fehler, korrigiere ihn, dann klicke erneut auf RUN.

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Bevor du die Seite von Babylon verlässt, kannst du dein Projekt speichern (SAVE). Die Seite ist später über den **Verlauf** deines Browsers wieder erreichbar. Oder du kopierst die Adresszeile in eine Textdatei und speicherst diese. Ich hebe mir meine Projekte auch gern als ZIP-Datei auf.

Zusammenfassung

Damit wäre erst einmal eine Verschnaufpause fällig. Nun hast du dich in Babylon schon ein wenig umgesehen und bist hoffentlich auf den Geschmack gekommen. Viel hast du noch nicht erreicht, aber du weißt schon einiges.

So kennst du ein paar Operationen im Umgang mit Babylon:

Babylon »starten«	Starte den Browser und wechsle auf die Babylon-Seite (*babylonjs.com*).
Ein Projekt öffnen	Klicke auf TRY , um zum »Playground« zu wechseln.
Ein Projekt bearbeiten	Ändere den Quelltext auf der linken Browserseite.
Ein Projekt starten	Klicke auf RUN und schau dir rechts das Ergebnis an.
Ein Projekt beenden	Verlasse die Playground-Seite.

Und du hast ein paar wichtige JavaScript-Wörter und -Zeichen gelernt:

`var`	Variable definieren
`function`	Funktion definieren
`return`	Funktion »abschließen«
`new`	Neues Objekt erzeugen
`()`	Klammern für die Funktion (Parameter)
`{}`	Klammern für den Anweisungsblock
`=`	Zuweisungszeichen
`//`	Kommentarzeichen
`;`	Anweisung abschließen

Einige davon haben weitreichendere Bedeutung (das wird später noch erklärt).

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Ein paar Fragen ...

1. Wie verändert man die Lage eines Objekts (zum Beispiel einer Kugel)?
2. Was ist der Unterschied zwischen 2D und 3D?
3. Wozu braucht man Vektoren?
4. Aus welchen Elementen kann oder muss ein Spiel mindestens bestehen?

... und eine Aufgabe

1. Erzeuge mehrere Objekte und positioniere sie auf dem Spielfeld. Füge die Zeilen dort im Quelltext ein, wo Kugel und Untergrund definiert sind:

Run | Save | Get .zip | New | Clear | Font: 12 | Safe Mode | -Editor | +Debug layer | Custom

```
1  var createScene = function () {
2      var scene = new BABYLON.Scene(engine);
3      var camera = new BABYLON.FreeCamera("camera1", new BABYLON.Vector3(0, 5, -10), scene);
4      camera.setTarget(BABYLON.Vector3.Zero());
5      camera.attachControl(canvas, true);
6      var light = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(0, 1, 0), scene);
7      light.intensity = 0.7;
8
9      var sphere = BABYLON.Mesh.CreateSphere("sphere1", 16, 2, scene);
10     sphere.position.y = 1;
11
12
13
14     var ground = BABYLON.Mesh.CreateGround("ground1", 12, 12, 2, scene);
15
16     return scene;
17 };
```

Forum | Sandbox | Wiki | Documentation

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Stichwortverzeichnis

! (Umkehr) 157
&& (Und) 159
|| (Oder) 159
1st Person 181
2D 27
3D 27
3rd Person 181

A

Absolut-Relativ 212, 244, 255
addEventListener 71, 100, 135, 144, 276
addToRenderList 229
Alignment 297
Alpha 92
ambientColor 93
Anatomie 242
Android
    bewegen 251
    drehen 255
    Figuren 248
    Gesicht 249
    Kollision 258
    Körper 243
Anführungszeichen 30
Animation
    Arme-Beine 268
    bone 273
    pause 276
    push 265
    restart 277
    Walking 271
AnimationType 264
Anweisung 24
Anzeige 295
applyGravity 182
applyImpulse 99
Array 169, 286
    getChildren 280
    push 265
atan2 146
attachControl 34, 124
Axis 256
Azimut 223

B

Babylon
    Camera 33
    Canvas 68
    Gravitation 182
    Impostor 96
    Kollision 182
    Light 34
    Material 88
    parent 232
    Physik 94
    Playground 76
    Scene 33
    Sprite 137
    Texture 88
    Texturen 230
Babylon JS 14, 22
babylon.js 48
Backface Culling 222
backFaceCulling 222
Bäume 210

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Bedingung 75
beginAnimation 266
Bildformat 87
Body 58, 62
Bone 273
Boolean 120
Browser 14
    Babylon 21
    Start aus Editor 58
Bumpmap 215
Bumpmapping 228
bumpTexture 215, 228

C

Camera 33
    Follow 253
    radius 254
cannon.js 95
Canvas 68
Canvas2D 295
cellIndex 163
checkCollisions 183, 288
childs 273
Chrome 90
clearColor 220
clearInterval 150, 302
clearTimeout 302
click 144, 158
clientX 285
clientX/clientY 144
clientY 285
collisionsEnabled 182, 187
Color 296
Compilation 38
const 171
createAndroid 246
createArea 104
CreateCylinder 190
createDisplay 301
createFollower 253
CreateGround 30
CreateGroundFromHeightMap 201
createHut 235
createObstacle 126, 285
CreatePlane 117
createPlayer 104
createScene 31, 67
CreateSphere 25
createTerrain 202, 208
createTree 211, 291
createWater 227

D

Debugging 324
delay 149
Dezimalpunkt 31
diffuseColor 92
diffuseTexture 88, 207
dispose 293
Drahtgestell 201
Du-Perspektive 181

E

Edge 58
Elastizität 99
Ellipsoid 182, 287
ellpsoidOffset 287, 289
else 83
else if 234
emissiveColor 93, 195
enablePhysics 94
Engine 14
Er/Sie-Perspektive 181
Ereignis 71, 100, 277
Ereignis-Funktion 100, 194
Error 38, 76
Esprima 324
Event 144
    click 144
    keydown 71, 135
    keyup 277
    mouse 284
    resize 69
EventListener 69

F

Face 249
Faktor 80
false 120
Farbe 296
Fehler 76, 122
Fehlermeldung 38
Fehlersuche 76
Fill 296

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

First Person 181
FollowCamera 253
fontName 297
for 129
FreeCamera 33, 122
friction 99
function 32, 69, 103
Funktion 31
    Aufruf 68, 107
    Definition 33
Funktionsanweisung 69

G

game.js 63
Gebäude 234
Gesicht 249
getAbsolutePosition 101
getChildMeshes 273
getFrontPosition 196
Gittermodell 201
gleich 75
Globale Variable 156
goToFrame 277
Gras 200
Gravitation 94, 97, 182
größer 75

H

hasAlpha 214
Head 46
Heightmap 203
heightOffset 254
Hexadezimal 296
Himmel 219
hit 145
hitPlayer 168
HTML
    Bedeutung 44
    body 58, 62
    head 46
    Kommentar 47
    meta 46
    script 46, 63
    src 48, 64, 116
    style 47
    title 46

I

Ich-Perspektive 181
Idle 267
if 72
if-else 150, 234
Impostor 96, 118, 204
index.html 43
Inklination 223
Intensity 34
Intersect 166
intersectsPoint 196
isPickable 165
isVisible 185

J

JavaScript 14, 23
JPG 87
JSLint 76, 324

K

Kamera
    Abstand 100
    Bindung 123
    erzeugen 33
    Position 122
    Steuerung 34, 124
    von oben 134
    Ziel 34, 123
Kegel 190
keyCode 72
keydown 71, 100, 135, 277
Keyframe
    Feld 265
    sammeln 265
keyup 277
Killed 165
Klammern
    eckig 169
    geschweift 32
    rund 32
Klang 300
kleiner 75
Knochen 273
Kollision 182, 257
    Bodenausgleich 289
    Ellipsoid 287

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by
mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Kommentar 23
Konstante 171, 286
Kontrollstruktur 72, 129
Koordinaten
    Ausgleich 256
Koordinatensystem 26
Kopf 243
Kosinus 258

L

Licht 238
Light 238
lockedTarget 123, 253
Loop 264

M

Masse 97
Material 86
    einsetzen 89
    erzeugen 88
Math
    PI 120
    random 128
Math.atan2 146
Math.cos 258
Math.random 157
Math.round 157
Math.sin 258
Math.sqrt 146
Matrix 270
Maus
    Ereignis 284
Mausklick 144
Mehrfach-Textur 207
Mesh 86, 200
MeshImpostor 204
Minus doppelt 72
Mixmap 207
mixTexture 208
mousemove 284
moveAndroid 251
moveLoop 149
movePlayer 104
moveWithCollisions 257

N

Negation 157
new 24, 36
Notepad++ 14, 51

O

Objekte 36
Objekt-Variable 33
oimo.js 95
onCollide 194, 290
Operationszeichen 128

P

paint.net 140
Parameter 32, 71, 103
    Klammerung 98
parent 211, 234, 243
Pfeiltasten 70
Physic Engine 94
PhysicsImpostor 96
Physik
    Gravitation 97
    Masse 97
Pi 120
pick 144, 162
pickedPoint 145
pickIntersect 166
pickResult
    hit 144
    pick 144
pickSprite 163
Pivot 269
Plane 117
Playground 22, 42, 76, 311
Plus doppelt 72
PNG 87
PointLight 238
position 26, 71
Prisma 190
Procedural 231
project.js 112
property 264
push 265
Pyramide 190
Pythagoras 146

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Q

Quadratwurzel 146
Quelltext 23

R

random 128, 157
    Bereich 170
Raute-Zeichen 296
Rectangle 295
Reibung 99
Relativ-Absolut 212, 244, 255
Rendern 69, 201
restitution 99
return 35, 106, 294
Rezept
    Animation 267
    Kollision 185
    Material 90
    Skybox 224
    Spielszene 35
    Sprite 139
    Terrainmaterial 209
    Wasser 230
RGB 92
Richtung 27
rotation 79
rotation (hut) 237
rotationOffset 254
round 157
Rumpf 243

S

scaling 79, 119
Schalt-Variable 120, 157
Script 46
Second Person 181
See 225
Semikolon 24
setAnimation 271
setInterval 149, 302
setKeys 265
setLimit 179
setLimits 118
setPivotMatrix 269
setSky 221
setTarget 34, 123
setTimeout 302
setVolume 300
shortcuts.xml 317
Sinus 258
Skalierung 119
Skalierungsfaktor 80
Skybox 221
SkyMaterial 221
sleep 149
Sonne-Erde 223
Sound 300
Source 64
Space.LOCAL 256
Space.WORLD 256
specular 134
specularColor 93, 214
speed 186
Sphere 25
Spielfeld 23
Spielfigur 23
Spielwelt 23
Sprite
    2D-Bild 137
    Bilderreihe 138
    cellIndex 163
    Transparenz 141
SpriteManager 138
sqrt 146
StandardMaterial 88
Start-Datei 44
stopAnimation 275

T

Tageszeiten 224
Tasten-Code 71
Tastenproblem 284
Terrain 200
terrain.js 210
TerrainMaterial 206
Textanzeige 295
Textur
    erzeugen 233
    nahtlos 87
    prozedural 231
    Terrain 207
Texturen 86
Third Person 181

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577

Timer 150, 302
Töne 300
Transformationen 79
translate 256
Translation 270
Transparenz 140
true 120

U

Umkehroperator 157
ungleich 75
uniqueId 195, 291
uScale 91

V

var 24, 32
Variable 72
    Funktion 33
    global 156
    Objekt 36
Vector3 37
Vektor 37
    Zero 100
Verknüpfungsoperator 159
vScale 91

W

Wasser 225
WaterMaterial 228
wav 300
Winkelfunktion 147, 258
Wireframe 87, 200, 205

X

x-Achse 26
xy-xz 145

Y

y-Achse 26

Z

z-Achse 27
Zählschleife 129
Zeitintervall 303
Zero 37, 100
Zufall 128, 157
Zuweisung 24
Zylinder 190

© des Titels »Spiele programmieren mit JavaScript für Kids« (ISBN 9783958455771) 2017 by
mitp Verlags GmbH & Co. KG, Frechen. Nähere Informationen unter: http://www.mitp.de/577