

Der **Offizielle**
ROBLOX
Guide

Games mit Roblox

Eigene Spiele entwickeln
mit Roblox Studio



Inhaltsverzeichnis

Vorwort	17
Über die Autorin	19
Über die weiteren Mitwirkenden	19
Über die Fachkorrektorin der deutschen Ausgabe	20
1 Was ist das Besondere an Roblox?	21
1.1 Roblox ermöglicht es, soziale Kontakte zu knüpfen	22
1.1.1 Roblox als soziale Website	23
1.1.2 Roblox als Treffpunkt für Entwickler	23
1.2 Roblox verwaltet Benutzerinhalte	24
1.2.1 Inhalte organisieren	25
1.2.2 Deine eigene Identität erschaffen	26
1.2.3 Charaktere personalisieren	27
1.3 Roblox ermöglicht schnelle Prototyperstellung und Iteration	29
1.3.1 Bereit für Anpassungen	29
1.4 Ideen einfach umsetzen	30
1.4.1 Plugins	31
1.4.2 Veröffentlichung und Updates ohne Wartezeit	31
1.5 Die Funktionen der Roblox-Engine	32
1.5.1 Vernetzung	32
1.5.2 Physik	33
1.5.3 Rendering	34
1.5.4 Plattformübergreifende Entwicklung	34
1.6 Alles kostenlos	35
1.7 Unbegrenzte Möglichkeiten	36
1.8 Finde deinen eigenen Stil	36
1.9 Zusammenfassung	38
1.9.1 Fragen und Antworten	38
1.9.2 Workshop	38
1.9.3 Aufgaben	39
2 Verwendung von Roblox Studio	41
2.1 Installation von Roblox Studio	41
2.1.1 Fehlerbehebung bei der Installation	42
2.1.2 Roblox Studio starten	43

2.2	Verwendung von Studio-Templates	44
2.2.1	Die Registerkarte All Templates	44
2.2.2	Themes	45
2.2.3	Gameplay	46
2.3	Verwendung des Game-Editors	47
2.3.1	Einrichtung des Arbeitsbereichs im Game-Editor	49
2.3.2	Verwendung des Explorer-Fensters	50
2.3.3	Erstellen eines Parts	51
2.3.4	Verwendung des Properties-Fensters	52
2.4	Verschiebung, Skalierung und Ausrichtung von Objekten	53
2.4.1	Verschieben	54
2.4.2	Skalieren	55
2.4.3	Drehen	56
2.4.4	Transformieren	57
2.5	Snapping	58
2.6	Collisions	58
2.7	Verankern	59
2.8	Speichern und Veröffentlichung deines Projekts	60
2.8.1	Speichern deines Projekts	60
2.8.2	Veröffentlichung deines Projekts	61
2.8.3	Projekt wieder öffnen	61
2.9	Spieltest	62
2.9.1	Spieltest durchführen	63
2.9.2	Spieltest beenden	63
2.10	Zusammenfassung	64
2.10.1	Fragen und Antworten	64
2.10.2	Workshop	65
2.10.3	Aufgaben	65
3	Verwendung von Parts	67
3.1	Erstellen eines Parts	67
3.2	Aussehen eines Parts ändern	67
3.2.1	Farbe	69
3.2.2	Material	69
3.2.3	Reflexionsgrad und Transparenz	71
3.3	Decals und Texturen erstellen	72
3.3.1	Decals	73
3.3.2	Texturen	76
3.4	Zusammenfassung	80
3.4.1	Fragen und Antworten	80
3.4.2	Workshop	80
3.4.3	Aufgaben	81

4	Physik-Engine	83
4.1	Verwendung von Attachments und Constraints	84
4.2	Erstellen einer Tür	86
4.3	CanCollide deaktivieren, damit ein Spieler die Tür durchschreiten kann	89
4.4	Scharniere und Federn hinzufügen	90
4.4.1	Öffnen einer Tür mit Scharnieren	90
4.4.2	Erstellen der Federn	94
4.4.3	Realistisches Verhalten von Federn	96
4.5	Verwendung eines Motors	98
4.6	Zusammenfassung	101
4.6.1	Fragen und Antworten	101
4.6.2	Workshop	101
4.6.3	Aufgaben	102
5	Landschaften gestalten	105
5.1	Der Terrain-Editor	106
5.2	Die Registerkarte Edit	109
5.2.1	Verändern des Terrains mit dem Substract-Werkzeug	111
5.2.2	Terrain anheben mit dem Grow-Werkzeug	112
5.2.3	Terrain abtragen mit dem Erode-Werkzeug	113
5.2.4	Terrain mit dem Smooth-Werkzeug glätten	113
5.2.5	Terrain mit dem Flatten-Werkzeug eibnen	114
5.2.6	Materialien ändern mit dem Paint-Werkzeug	115
5.2.7	Wasserflächen erzeugen mit dem Sea-Level-Werkzeug	117
5.3	Die Registerkarte Region	118
5.3.1	Terrain auswählen	119
5.3.2	Terrain mit dem Move-Werkzeug verschieben	120
5.3.3	Terrain mit dem Resize-Werkzeug skalieren	122
5.3.4	Verwendung der Werkzeuge Copy, Paste und Delete	123
5.3.5	Eine Fläche mit dem Fill-Werkzeug ausfüllen	123
5.4	Height Maps und Color Maps	124
5.4.1	Height Maps	124
5.4.2	Color Maps	126
5.5	Zusammenfassung	127
5.5.1	Fragen und Antworten	127
5.5.2	Workshop	128
5.5.3	Aufgaben	129

6	Beleuchtung	131
6.1	Eigenschaften der Beleuchtung	132
6.1.1	Eigenschaften des Erscheinungsbilds	134
6.1.2	Die Eigenschaften Data und Exposure	136
6.2	Beleuchtungseffekte	137
6.2.1	SpotLight, PointLight und SurfaceLight	140
6.3	Zusammenfassung	143
6.3.1	Fragen und Antworten	143
6.3.2	Workshop	143
6.3.3	Aufgaben	144
7	Atmosphäre	147
7.1	Eigenschaften der Atmosphäre	148
7.1.1	Density	149
7.1.2	Offset	150
7.1.3	Haze	151
7.1.4	Color	152
7.1.5	Glare	153
7.1.6	Decay	154
7.2	Anpassen der Skybox	156
7.2.1	Erstellen einer Skybox	156
7.2.2	Anpassen der Himmelskörper	159
7.2.3	Anpassen der Beleuchtungsfarben	160
7.3	Zusammenfassung	162
7.3.1	Fragen und Antworten	162
7.3.2	Workshop	162
7.3.3	Aufgabe	163
8	Effekte	165
8.1	Partikel	165
8.1.1	Anpassen der Partikel	167
8.1.2	Ändern der Farbe von Partikeln	168
8.1.3	Eigenschaften eines ParticleEmitters	169
8.2	Beams	170
8.2.1	Krümmung	172
8.2.2	Segments	174
8.2.3	Width	175
8.2.4	Lichtern mit einem Beam einen Strahleneffekt hinzufügen	176

8.3	Zusammenfassung	177
8.3.1	Fragen und Antworten	177
8.3.2	Workshop	178
8.3.3	Aufgaben	178
9	Objekte importieren	183
9.1	Einfügen und Hochladen kostenloser Modelle	183
9.1.1	Hochladen des Modells bei Roblox	185
9.1.2	Zugriff auf Modelle	187
9.1.3	Einfügen kostenloser Modelle	187
9.2	Importieren mit MeshParts und Asset Manager	189
9.2.1	Mehrere Meshes auf einmal importieren mit dem Asset Manager	192
9.3	Importieren von Texturen	194
9.3.1	Importieren von Decals mit dem Asset Manager	196
9.4	Sounds importieren	196
9.5	Zusammenfassung	197
9.5.1	Fragen und Antworten	198
9.5.2	Workshop	198
9.5.3	Aufgaben	199
10	Spielstruktur und Zusammenarbeit	201
10.1	Einem Spiel Places hinzufügen	201
10.2	In Roblox Studio zusammenarbeiten	203
10.2.1	Zusammenarbeit bei Gruppenspielen	204
10.2.2	Konfiguration der Rollen	204
10.2.3	Zuweisung der Rollen	205
10.2.4	Team Create aktivieren	205
10.2.5	Hinzufügen und Verwalten von Benutzern in Team Create	206
10.2.6	Zugriff auf die Team-Create-Sitzung	208
10.2.7	Verwendung von Roblox Studio Chat	209
10.2.8	Team Create deaktivieren	209
10.3	Erstellen von Roblox-Packages und darauf zugreifen in Roblox Studio	210
10.3.1	Objekte in Packages konvertieren	210
10.3.2	Zugriff auf die Package-Toolbox	212
10.3.3	Zugriff auf Packages im Asset Manager	212
10.3.4	Aktualisieren eines Packages	213
10.3.5	Eine größere Anzahl von Packages aktualisieren	214

10.4	Zusammenfassung	215
10.4.1	Fragen und Antworten	216
10.4.2	Workshops	216
10.4.3	Aufgaben	216
11	Lua – ein Überblick	219
11.1	Der Coding Workspace	220
11.1.1	Erstellen des ersten Scripts	220
11.2	Eigenschaften ändern mithilfe von Variablen	222
11.2.1	Überblick über Variablen	222
11.2.2	Verwendung von Variablen	223
11.2.3	Erstelle eine halb durchsichtige Bombe	223
11.3	Kommentare zum Code hinzufügen	224
11.4	Funktionen und Events	226
11.4.1	Erstellen einer Funktion	226
11.4.2	Verwendung einer Funktion, um eine Bombe explodieren zu lassen	227
11.4.3	Der Einsatz von Events	228
11.4.4	Verwendung eines Events, um einen Part explodieren zu lassen, wenn er berührt wird	228
11.5	Verwendung bedingter Anweisungen	229
11.6	Arrays und Dictionarys	230
11.7	Verwendung von Schleifen	231
11.8	Gültigkeitsbereiche	234
11.9	Benutzerdefinierte Events	235
11.10	Debugging des Codes	237
11.10.1	Debugging mit Strings	237
11.10.2	Lua-Debugger	237
11.10.3	Log-Dateien	238
11.11	Zu einem besseren Spieleentwickler werden	239
11.12	Zusammenfassung	240
11.12.1	Fragen und Antworten	240
11.12.2	Workshop	241
11.12.3	Aufgaben	241
12	Kollisionen, Humanoide, Punktzahl	243
12.1	Kollisionen	243
12.1.1	CollisionFidelity	243
12.1.2	Anzeige und Verbesserung der Kollisionsgeometrie	244
12.1.3	Collision Groups Editor	245
12.1.4	Direkte Verwendung des Collision Groups Editors	246
12.1.5	Verwendung des Collision Group Editors per Script	247

12.2	Kollisionserkennung	249
12.2.1	Verwendung von Touched	249
12.2.2	debounce	250
12.3	Humanoid	253
12.3.1	Humanoid innerhalb der Hierarchie	253
12.3.2	Eigenschaften, Funktionen und Events	254
12.4	Zusammenfassung	262
12.4.1	Fragen und Antworten	262
12.4.2	Workshop	263
12.4.3	Aufgaben	263
13	Interaktion mit der Benutzeroberfläche (GUI)	265
13.1	Erstellen von GUIs	266
13.1.1	PlayerGui	266
13.1.2	SurfaceGui	270
13.2	Grundlegende GUI-Elemente	274
13.3	Programmierung interaktiver GUIs	274
13.4	Tweening	277
13.5	Layouts	278
13.6	Erstellen einer GUI mit Countdown	282
13.7	Zusammenfassung	283
13.7.1	Fragen und Antworten	283
13.7.2	Workshop	284
13.7.3	Aufgaben	285
14	Programmierung von Animationen	287
14.1	Position und Rotation	287
14.1.1	Bewegen eines Objekts von Punkt A nach Punkt B	289
14.1.2	Rotation von Parts mit CFrames	291
14.2	Ruckelfreies Bewegen von Objekten mit Tween	295
14.2.1	Tweening zwischen zwei Punkten	296
14.2.2	EasingStyle und EasingDirection	297
14.3	Bewegen des gesamten Modells	298
14.4	Zusammenfassung	300
14.4.1	Fragen und Antworten	300
14.4.2	Workshop	300
14.4.3	Aufgaben	301

15	Sounds und Musik	303
15.1	Einen Soundtrack erstellen	303
15.2	Importieren von Musik und Sounddateien	305
15.3	Umgebungsgeräusche hinzufügen	306
15.4	Sounds via Code abspielen	308
15.5	Gruppieren von Sounds	309
15.6	Zusammenfassung	311
15.6.1	Fragen und Antworten	311
15.6.2	Workshop	312
15.6.3	Aufgaben	313
16	Animation-Editor	315
16.1	Einführung in den Animation-Editor	316
16.1.1	Anforderungen an das Modell	316
16.1.2	Öffnen des Animation-Editors	317
16.2	Erstellen von Posen	318
16.3	Speichern und Exportieren von Animationen	322
16.4	Easing	323
16.5	Inverse Kinematik	324
16.5.1	IK aktivieren	324
16.5.2	Parts anheften	326
16.6	Einstellungen für Animationen	327
16.6.1	Wiederholungen	327
16.6.2	Priorität	328
16.7	Animation-Events	328
16.7.1	Events hinzufügen	329
16.7.2	Verschieben und Löschen von Events	330
16.7.3	Events klonen	330
16.7.4	Implementierung von Events in Scripts	330
16.7.5	Standardanimationen ersetzen	332
16.8	Zusammenfassung	333
16.8.1	Fragen und Antworten	334
16.8.2	Workshop	334
16.8.3	Aufgaben	335
17	Kämpfe, Teleportation und Datenspeicher	337
17.1	Einführung in Tools	337
17.1.1	Grundlagen	338
17.1.2	Ein Tool erstellen	339
17.1.3	Tool-Handle	339
17.1.4	Tool-Ausrichtung	341

17.2	Teleportation	347
17.2.1	Teleportation innerhalb eines Place	348
17.2.2	Teleportieren zwischen verschiedenen Places	350
17.2.3	Spieluniversen	350
17.3	TeleportService	351
17.3.1	Funktionen	351
17.3.2	Abrufen der PlaceID	352
17.3.3	Beispiel: Client	352
17.3.4	Beispiel: Server	353
17.4	Verwendung des dauerhaften Datenspeichers	356
17.4.1	Unterstützte Datentypen und Beschränkungen	356
17.5	Funktionen des Datenspeichers	360
17.5.1	UpdateAsync() und SetAsync()	362
17.6	Fehlerbehandlung	362
17.6.1	Was ist ein pcall?	362
17.6.2	Schutz vor Datenverlust	363
17.7	Zusammenfassung	363
17.7.1	Fragen und Antworten	363
17.7.2	Workshop	364
17.7.3	Aufgaben	364
18	Mehrspieler-Code und das Client-Server-Modell	367
18.1	Das Client-Server-Modell	367
18.1.1	Scripts und LocalScripts	368
18.1.2	Replikation	368
18.2	Was sind RemoteFunctions und RemoteEvents?	369
18.2.1	Verwendung von RemoteEvent und RemoteFunction	370
18.2.2	Erstellen eines RemoteEvents	371
18.3	Serverseitige Validierung	374
18.4	Teams	375
18.4.1	Hinzufügen von Teams	375
18.4.2	Automatische Zuordnung von Spielern zu einem Team	376
18.4.3	Manuelle Zuordnung von Spielern zu einem Team	377
18.5	Network-Ownership	378
18.6	Zusammenfassung	378
18.6.1	Fragen und Antworten	379
18.6.2	Workshop	379
18.6.3	Aufgaben	380

19	Module-Scripts	383
19.1	Kurz vorgestellt: Das Module-Script	383
19.1.1	Aufbau eines Module-Scripts	384
19.1.2	Hinzufügen von Code, der überall verwendet werden kann	385
19.1.3	Verwendung eines Module-Scripts	385
19.2	Clientseitige und serverseitige Module-Scripts	387
19.3	Der Einsatz von Module-Scripts: Game-Loop	389
19.3.1	Einstellungen	390
19.3.2	Erstellen wiederverwendbarer Funktionen für Spielrunden	391
19.3.3	Die Game-Loop	391
19.4	Zusammenfassung	394
19.4.1	Fragen und Antworten	394
19.4.2	Workshop	394
19.4.3	Aufgaben	395
20	Programmierung von Kamerabewegungen	397
20.1	Einführung in die Verwendung von Kameras	397
20.1.1	Kameraeigenschaften	399
20.1.2	Handhabung der Kamera	400
20.2	Programmierung einer Kamerabewegung	401
20.3	Verwendung von Render-Step	402
20.4	Versetzen der Kamera	403
20.4.1	Dauerhafte Verknüpfung mit dem Render-Step	406
20.4.2	deltaTime	408
20.5	Zusammenfassung	409
20.5.1	Fragen und Antworten	409
20.5.2	Workshop	409
20.5.3	Aufgaben	410
21	Plattformübergreifende Entwicklung	413
21.1	Verbesserung der Performance	413
21.1.1	Speicherbedarf	413
21.1.2	Optimierung	414
21.1.3	Vereinfachung des physikalischen Verhaltens	416
21.1.4	Inhalte streamen	416
21.1.5	Diverse weitere Optimierungen	417
21.2	Verbesserung der Scripts	418
21.2.1	Zuweisung des Parents bei Objekten	418
21.2.2	Blindes Vertrauen in Server/Client	419
21.2.3	Schleifen sparsam verwenden	419

21.3	Kompatibilität mit Mobilgeräten	420
21.3.1	Erscheinungsbild	420
21.3.2	Steuerung	421
21.3.3	Simulation von Mobilgeräten	422
21.4	Spielkonsolen und VR	424
21.4.1	Xbox-Richtlinien	424
21.4.2	VR Best Practices	425
21.5	Zusammenfassung	425
21.5.1	Fragen und Antworten	426
21.5.2	Workshop	426
21.5.3	Aufgaben	427
22	Globale Community	429
22.1	Einführung in Lokalisierung	429
22.1.1	Texte für die Übersetzung erfassen	429
22.1.2	Übersetzung der erfassten Texte	431
22.1.3	Einsetzen der Übersetzung	432
22.2	Globale Regelkonformität	433
22.3	Datenschutzgesetze: DSGVO und CCPA	434
22.3.1	Allgemeine Richtlinien	435
22.3.2	Löschen von Spielerdaten	435
22.4	Zusammenfassung	437
22.4.1	Fragen und Antworten	437
22.4.2	Workshop	438
22.4.3	Aufgaben	439
23	Monetarisierung	441
23.1	Game Pass: Einmalige Käufe	441
23.2	Game Passes im Spiel verkaufen	443
23.2.1	Game-Pass-Vorteile aktivieren	444
23.3	Developer Products: Consumables	445
23.4	Roblox Premium	448
23.5	Developer Exchange: Verdienne richtiges Geld mit deinem Spiel ...	450
23.6	Zusammenfassung	452
23.6.1	Fragen und Antworten	452
23.6.2	Workshop	453
23.6.3	Aufgaben	454
24	Spieler auf das Spiel aufmerksam machen	457
24.1	Icons, Vorschaubilder und Trailer	457
24.2	Updates	461

24.3	Anzeigen und Benachrichtigungen	461
24.3.1	Sponsoranzeigen	461
24.3.2	Benutzeranzeigen	464
24.3.3	Benachrichtigungen	466
24.4	Analytics	468
24.5	Zusammenfassung	469
24.5.1	Fragen und Antworten	469
24.5.2	Workshop	469
24.5.3	Aufgaben	470
A	Lua-Scripting	473
A.1	Änderung von Eigenschaften (Datentyp und Enumerationen)	473
A.2	Bedingte Anweisungen und Verzweigungen	474
A.3	Ausbau der Lua-Kenntnisse	476
B	Eigenschaften und Funktionen von Humanoid	477
	Stichwortverzeichnis	481



Vorwort

Stelle dir ein virtuelles Universum vor, das von einer weltweiten Community entwickelt wurde, der Künstler, Programmierer, Geschichtenerzähler und die verschiedensten anderen Leute angehören. In diesem Traum würden Menschen aus allen Teilen der Welt zusammenkommen, um Millionen Erlebnisse zu erschaffen und mit ihren Freunden zu teilen und um voneinander zu lernen. Es wäre ein Universum, das auf der Vorstellungskraft beruht, in dem alles möglich wäre, unabhängig vom verwendeten Gerät, dem Aufenthaltsort oder der Zeit. Und wenn ich nun feststelle, dass es diese digitale Utopie schon seit mehr als einem Jahrzehnt gibt?

Als Erik Cassel und ich 2004 Roblox mitgründeten, wollten wir einen immersiven, dreidimensionalen, physikalisch simulierten Raum für mehrere Spieler erschaffen, mit dem sich jeder verbinden kann, um zusammen mit anderen Spaß zu haben. In den Anfangstagen von Roblox waren wir davon fasziniert, was die Leute alles anstellten. Sie wollten ihr eigenes Restaurant managen, eine Naturkatastrophe überleben oder erfahren, wie es ist, ein Vogel zu sein. Wenn ich nun, siebzehn Jahre später, in die Zukunft blicke, ist es offensichtlich, dass diese Plattform noch viel mehr zu bieten hat.

Roblox stellt eine neue Kategorie für Erlebnisse mit anderen Menschen dar, in der die Grenzen zwischen Spiel, sozialem Netzwerk und Medien verschwimmen. Unser Team hat festgestellt, dass sich die Millionen täglichen Roblox-User nicht nur zum Spielen anmelden, sondern sich zusammenfinden, um eine Community aufzubauen, Geschichten zu erzählen und sowohl mit Freunden als auch mit Fremden etwas zu erleben.

Wir fahren damit fort, unsere Plattform weiterzuentwickeln, die es Milliarden von Benutzern ermöglicht, Erfahrungen mit anderen zu teilen – es gab nie einen besseren Zeitpunkt, einer weltweiten Community kreativer Menschen beizutreten, die so erstaunliche Beiträge zu unserer Plattform leisten. 3-D-Erlebnisse zu entwickeln, macht nicht nur Spaß, sondern bietet auch die Möglichkeit, Fähigkeiten und Kenntnisse zu sammeln, die für eine Laufbahn in den Bereichen Informatik, Design, Kunst und vielen anderen erforderlich sind. Viele der besten Entwickler unserer Plattform haben das Geld, das sie mit Roblox verdient haben, verwendet, um ihr Studium zu finanzieren, ein eigenes Studio für Spieleentwicklung zu gründen oder um eine Anzahlung für das Haus ihrer Eltern zu leisten.

Ich bin davon überzeugt, dass Roblox letzten Endes zur Entstehung des *Metaverse* führt, einer allumfassenden digitalen Realität, die durch unsere physische ergänzt wird. Wir sehen den Tag kommen, an dem die Menschen Roblox nicht nur aufsuchen, um zu spielen und Kontakte zu knüpfen, sondern auch, um geschäftliche Treffen abzuhalten oder die Schule zu besuchen. Die Möglichkeiten, die das *Metaverse* bietet, nehmen mit jedem Tag zu. Das gilt auch für den Bedarf an innovativen und kreativen Entwicklern, die für die Erlebnisse sorgen, von denen wir in der Science-Fiction seit Jahren träumen.

Ich lade dich herzlich dazu ein, an der Welt von Roblox teilzunehmen, nicht nur als Spieler, sondern auch als Entwickler. Zu lernen, Spiele und immersive 3-D-Erlebnisse zu entwerfen, trägt dazu bei, weltweit Millionen von Menschen durch die Spiele zu verbinden und eine Community aufzubauen, die nicht durch Grenzen, Sprachen oder Geografie definiert ist. Wenn du am Programmieren, an Game Design oder der immersiven 3-D-Welt von Roblox interessiert bist, solltest du dieses Buch lesen und deine verrücktesten und kreativsten Ideen verfolgen. Das *Metaverse* braucht Entwickler wie dich.

Deine Fantasie wartet schon!

*David »Builderman« Baszucki
Gründer und Geschäftsführer
Roblox Corporation*

Über die Autorin



Genevieve Johnson ist als Senior Instrucional Designer bei Roblox tätig, der weltweit größten sozialen Spiele-Plattform mit benutzergenerierten Inhalten. Sie betreut die Erstellung von Bildungsinhalten und berät Pädagogen rund um den Globus dabei, wie man Roblox in STEAM-basierten Lernprogrammen einsetzen kann. Ihre Arbeit ermöglicht es Schülern, Laufbahnen als Unternehmer, Ingenieur und Designer einzuschlagen.

Bevor sie zu Roblox kam, war Johnson für bildungsrelevante Inhalte bei ID Tech verantwortlich, einem technischen Ausbildungsprogramm, das jährlich mehr als 50.000 Schülerinnen und Schüler zwischen 6 und 18 Jahren nutzen. Bei ID Tech war sie an der Einrichtung eines erfolgreichen STEAM-Programms nur für Mädchen beteiligt. Ihr Team hat Ausbildungsinhalte für mehr als 60 technologiebezogene Kurse für eine Vielfalt von Themen entwickelt, von der Programmierung über Robotik bis hin zum Game Design.

Über die weiteren Mitwirkenden

Ashan Sarwar ist ein Roblox-Entwickler, der seit 2013 Roblox Studio verwendet. Von ihm ist *LastShot*, ein Shooter, der auf Roblox verfügbar ist.

Raymond Zeng ist ein Roblox-Entwickler, der das Programmieren liebt und andere Programmierer auf jedem Niveau unterrichtet. Unter der Bezeichnung *MacAndSwiss* betreibt er einen YouTube-Kanal, auf dem er Lua unterrichtet, über Neuigkeiten zu Roblox berichtet und seine Programmierprojekte vorstellt.

Theo Docking ist seit vier Jahren als Spieleprogrammierer tätig. Er mag es, an spannenden Projekten zu arbeiten, die Roblox richtig ausreizen und dabei interessante Menschen zu treffen. Außerdem liebt er es, mit Roblox' Physik-Engine zu experimentieren und Code für NFCs, Autos und mehr zu schreiben. Wenn er nicht programmiert, zeichnet er Pläne für neue Spiele oder spielt *Ultimate Driving*, um sich zu entspannen.

Joshua Wood hat Roblox 2013 entdeckt und ein Jahr später angefangen, seine eigenen Spiele zu erstellen. Er ist der Entwickler von *Game Dev Life*, das mehr als eine Million mal gespielt wurde. Zudem ist er Entwickler von *DoubleJGames*.

Swathi Sutrave ist ein selbsternannter Technik-Geek. Sie war als Expertin für verschiedene Programmiersprachen, unter anderem auch Lua, für Unternehmen, Start-ups und Universitäten tätig.

Henry Chang ist ein Computergrafikdesigner, der sich verschiedener Medien bedient, unter anderem zwei- und dreidimensionaler Grafiken und Animationen. Er ist Autodidakt und hat verschiedene Möglichkeiten interaktiver Medien ausgelotet. Weitere Informationen sind unter <https://www.henrytcgweb.com> zu finden.

Über die Fachkorrektorin der deutschen Ausgabe



Tanja Köhler hat Architektur studiert und war für verschiedene Architekturbüros im In- und Ausland tätig. Als 3D-Artistin vertiefte sie ihre 3D-Software-Kenntnisse in der Computerspiele-Branche bei Opus Studio Inc. (Tokio) und Bluebyte/Ubisoft. Unter anderem visualisierte sie die Gebäude für *Die Siedler VI – Aufstieg eines Königreiches*. Heute lebt und arbeitet sie als freie Architektin bei München und ist Lehrbeauftragte an der TU München. Dort unterrichtet sie Studierende der Architekturinformatik und Games Engineering. Zudem betreibt sie ein Studio für Programmierung und Computerspiele-Design für Kinder und Jugendliche; weitere Informationen unter www.nextlevel-gauting.de.

Verwendung von Roblox Studio

Die Themen in diesem Kapitel:

- Installation und Start von Roblox Studio
- Wie man Studio-Templates (Vorlagen) verwendet
- Navigation im Game-Editor
- Wie man einen Part erstellt
- Parts verschieben, skalieren und ausrichten
- Projekte speichern und veröffentlichen
- Spieltests

Jetzt hast du die Unternehmenskultur und die Features erkundet, die Roblox zu etwas Besonderem machen, und kannst deiner Kreativität mit der kostenlosen Spiel-Engine *Roblox Studio* freien Lauf lassen. Roblox Studio ist ein Spielplatz für Entwickler, auf dem sie ihre Spiele auf der Roblox-Website erstellen, teilen und spielen können. Das Tolle an dieser Plattform ist, dass du alle möglichen Umgebungen ganz einfach erstellen kannst, sei es eine Vulkaninsel oder eine Stadtlandschaft. Anschließend kannst du einen Charakter in dieser Welt platzieren und sofort mit dem Spielen loslegen. Stell dir einen riesengroßen Spielplatz vor, auf dem alle Werkzeuge vorhanden sind, um fantastische Welten zu erstellen – das ist Roblox Studio.

In diesem Kapitel erfährst du, wie Roblox Studio installiert wird und wie du Roblox-Templates verwendest. Außerdem lernst du, wie du deinen Workspace (Arbeitsbereich) einrichtest, damit er in der 3-D-Welt Objekte aufnehmen kann. Du erfährst, welche Unterschiede es zwischen dem Speichern und der Veröffentlichung eines Projekts gibt. Und zum Abschluss lernst du, wie du dein Spiel vor der Veröffentlichung testen kannst.

2.1 Installation von Roblox Studio

Roblox Studio ist eine kostenlose und immersive Plattform für Spieleentwickler, mit der verschiedene Terrains, Städte, Gebäude, Rennspiele und vieles mehr erstellt werden können. Du benötigst keine jahrelange Programmiererfahrung oder

einen Abschluss, um tolle Spiele zu erstellen. Alles, was du brauchst, sind deine Fantasie und praktische Erfahrung im Umgang mit Roblox Studio. Roblox Studio ist äußerst intuitiv bedienbar und da es plattformübergreifend funktioniert, können Entwickler es sowohl auf Windows- als auch auf Mac-Systemen installieren.

Führe zur Installation von Studio die folgenden Schritte aus:

1. Rufe <https://www.roblox.com/create> auf.
2. Klicke auf **ERSTELLE ETWAS** und in dem dann angezeigten Fenster auf **STUDIO HERUNTERLADEN**.
3. Öffne den Ordner, in den Studio heruntergeladen wurde und doppelklicke auf die Datei, um es zu installieren.

Systemanforderungen

Damit Roblox Studio effizient arbeitet, müssen Betriebssystem und Hardware bestimmte Voraussetzungen erfüllen:

- Roblox Studio läuft nicht auf Linux, Chromebooks oder mobilen Geräten, wie etwa Smartphones.
- Auf einem Windows-PC muss mindestens Windows 7 installiert sein; auf einem Mac ist mindestens macOS 10.10 erforderlich.
- Minimal 1 GB Systemarbeitsspeicher ist erforderlich.
- Zum Herunterladen und zur Aktualisierung ist ein Internetzugang notwendig. Darüber hinaus bietet dieser dir die Möglichkeit, Projekte in deinem Roblox-Konto zu speichern (veröffentlichen).

Studio lässt sich noch besser nutzen, wenn Folgendes verfügbar ist (nicht zwingend notwendig):

- Eine Maus mit Scrollrad, am besten eine Maus mit drei Tasten
- Eine eigenständige Grafikkarte (keine integrierte Grafikkarte)

2.1.1 Fehlerbehebung bei der Installation

Wenn du die erforderlichen Schritte zur Installation durchgeführt hast, es aber Hardwarekonflikte gibt, solltest du zur Fehlerbehebung Folgendes überprüfen:

- Wenn du kürzlich neue Hardware oder Treiber installiert hast, solltest du diese entfernen oder ersetzen, um festzustellen, ob sie das Problem verursachen.
- Verwende eine Diagnose-Software und suche nach Informationen zur Fehlerbehebung des Betriebssystems.
- Starte den Computer neu.
- Deinstalliere und lösche alle Roblox-Dateien und installiere die neueste Version von Studio erneut.

Wenn es immer noch zu Fehlern kommt, kannst du dich auch an das Roblox-Support-Forum wenden, um weitere Tipps zu erhalten.

2.1.2 Roblox Studio starten

Nach der erfolgreichen Installation von Roblox Studio kannst du es starten.

1. Doppelklicke auf das Programmsymbol auf dem Desktop (Windows) oder klicke auf das Programmsymbol im Dock (Mac), um ein Anmeldefenster zu öffnen (Abbildung 2.1).
2. Gib deinen Benutzernamen und dein Passwort ein.
3. Klicke auf **LOG IN**.

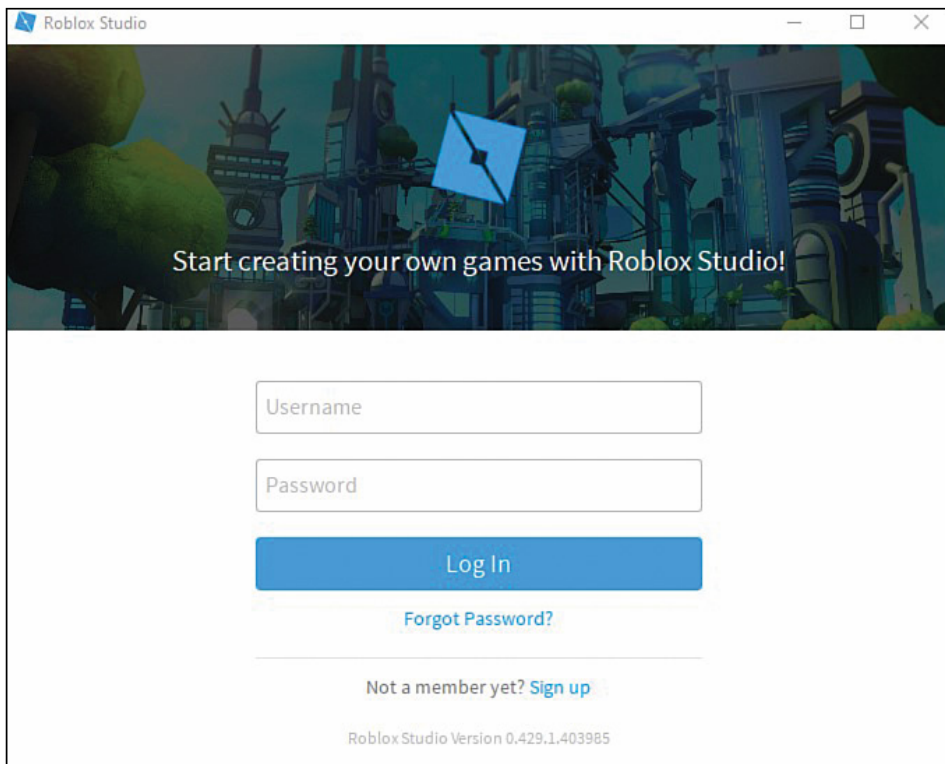


Abb. 2.1: Anmeldefenster von Roblox Studio

Nach erfolgter Anmeldung wird eine Seite mit verschiedenen Templates und eine Seitenleiste mit den Menüpunkten **NEW**, **MY GAMES**, **RECENT** und **ARCHIVE** angezeigt (Abbildung 2.2).

Der folgende Abschnitt enthält eine kurze Einführung in die Verwendung dieser Templates und den anderen Teilen von Studio. Anschließend kannst du anfangen, mit den Werkzeugen von Studio zu experimentieren.

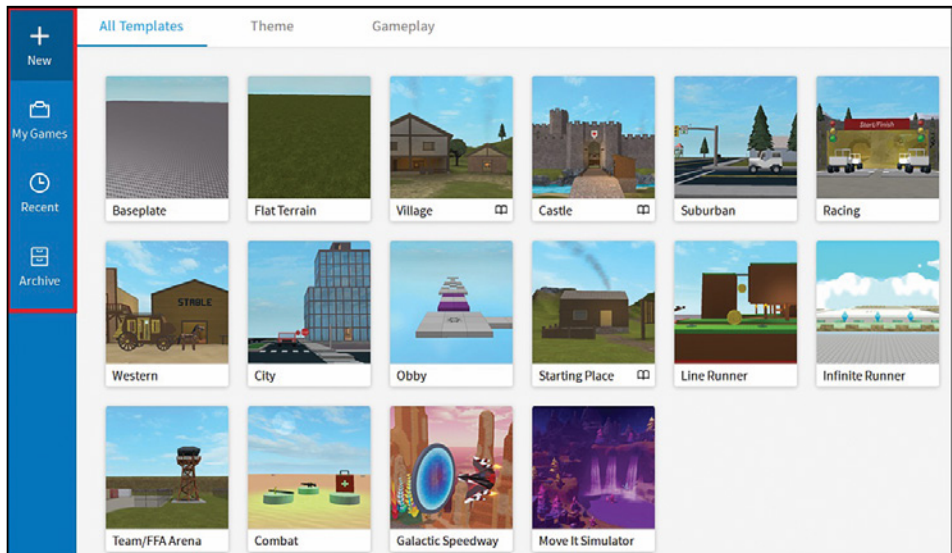


Abb. 2.2: Startbildschirm von Roblox Studio

2.2 Verwendung von Studio-Templates

Wenn du Roblox Studio zum ersten Mal startest, werden unter dem Menüpunkt **NEW** der Seitenleiste drei Registerkarten angezeigt: **ALL TEMPLATES**, **THEME** und **GAMEPLAY**. Templates sind vorgefertigte Projekte, die du dir beim Aufbau deiner eigenen Spielewelt zum Vorbild nehmen kannst.

2.2.1 Die Registerkarte All Templates

ALL TEMPLATES (Abbildung 2.3) ist eine Kombination der Registerkarten **THEME** und **GAMEPLAY**. Du kannst diese Templates als Ausgangspunkt für deine Spiele verwenden. Wenn du beispielsweise ein mittelalterliches Spiel entwickelst: Das Theme *Castle* (Schloss oder Burg) enthält eine Vielzahl dazu passender Details. Oder wenn du ein interaktives Obby (eine Art Parkour mit Hindernissen, die überwunden werden müssen) entwickeln möchtest, kannst du das Gameplay-Template *Obby* verwenden. Zwei einfache Templates sind allgemein gut für den Start geeignet:

- **Baseplate** (Grundplatte): Wird gerne gewählt, um direkt loszulegen. Die Baseplate selbst kann leicht entfernt werden, sodass eine leere Leinwand verbleibt, die man bearbeiten kann.
- **Flat Terrain** (ebenes Gelände): Dieses Template besitzt statt einer Baseplate eine flache grasbedeckte Ebene. Mit dem Terrain-Editor kannst du das Gelände bearbeiten oder entfernen.

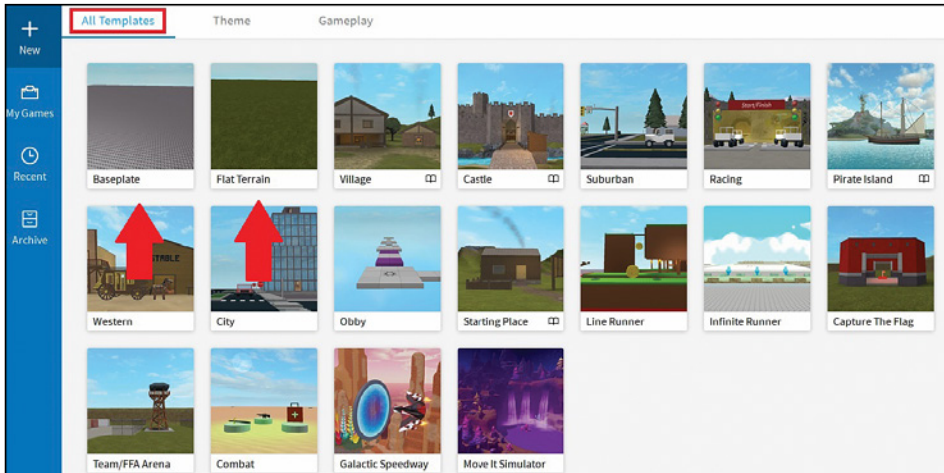


Abb. 2.3: Auf dem Startbildschirm von Roblox Studio werden verschiedene verfügbare Templates angezeigt, beispielsweise die Templates BASEPLATE und FLAT TERRAIN.

2.2.2 Themes

Themes sind eine Kombination aus Gameplay und Objekten und bilden zusammen eine neue Welt. Sie verleihen deinem Spiel eine gewisse Stimmung. Bei einer Schlacht im Weltraum wird es beispielsweise Asteroiden und andere galaktische Komponenten geben. Roblox stellt einige vorgefertigte Themes zur Verfügung, die sofort einsatzbereit sind, aber nach Belieben verändert werden können. Wenn du eine Spielwelt erkundest, stößt du auf Beschreibungen, die auf Anwendungsfälle oder Features hinweisen. Dazu gehören auch Tipps, die erläutern, wie die Effekte erzielt werden – falls du diese selbst umsetzen möchtest.

Village (Dorf) ist ein Beispiel für ein vorgefertigtes Theme (Abbildung 2.4). Du kannst die Häuser erkunden und die Wege im Ort beschreiten, die dich zu einem Fluss, einer Brücke und schließlich zu einer Werft führen, von der aus du kleine Inseln sehen kannst.



Abb. 2.4: VILLAGE ist ein Beispiel für eines der vorgefertigten Themes, die in Studio verfügbar sind.

2.2.3 Gameplay

Einige Templates enthalten interaktive Spielkomponenten. Dazu gehören beispielsweise Team Deathmatch, Control Points, Capture the Flag und andere (Abbildung 2.5). An diesen Templates ist besonders praktisch, dass Entwickler sie zerlegen und gezielt bestimmte Komponenten entnehmen können, beispielsweise die Verwendung eines Radars oder der Startpunkte für Teams. Diese Templates bieten Komponenten, die es ermöglichen, festzulegen, was einem Spieler in einem Spiel möglich ist, was die Ziele des Spiels sind und wie das Spiel modifiziert werden kann.

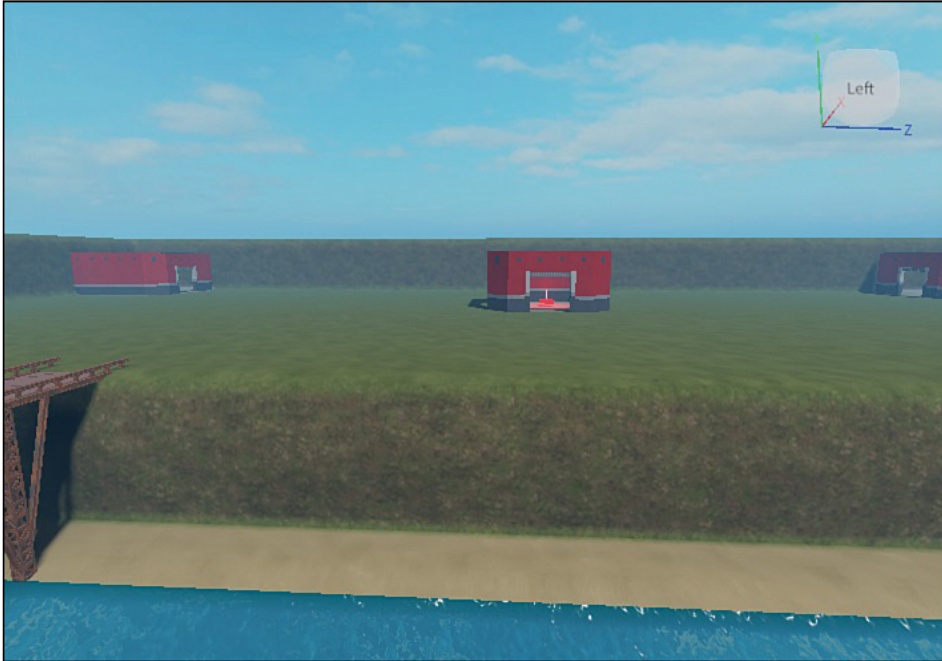


Abb. 2.5: *Capture the Flag*: Beispiel für ein vorgefertigtes Gameplay-Template

2.3 Verwendung des Game-Editors

Jetzt bist du mit der Startseite von Studio schon vertraut. Klicke als Erstes auf das Template BASEPLATE. Dadurch wird der Game-Editor gestartet (Abbildung 2.6).

Wie der Name schon sagt, dient der Game-Editor dazu, Spiele zu erstellen, zu bearbeiten und zu testen. Am oberen Ende des Game-Editors befindet sich eine Menüleiste mit mehreren Registerkarten (Abbildung 2.7).

Kapitel 2

Verwendung von Roblox Studio

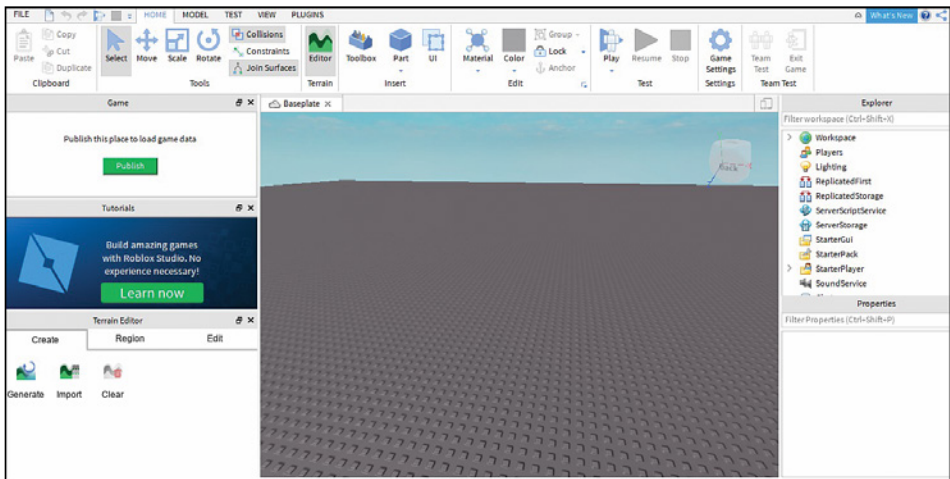


Abb. 2.6: Der Game-Editor ermöglicht es dir, dein Spiel zu erstellen, zu bearbeiten und zu testen.



Abb. 2.7: Die Menüleiste von Roblox Studio

- **HOME:** Eine Registerkarte, auf der kurz und übersichtlich alle Features enthalten sind, die häufig genutzt werden, sodass man schnell und einfach darauf zugreifen kann.
- **MODEL:** Verfügt neben *move* (bewegen), *scale* (Größe ändern) und *rotate* (drehen) über weitere Werkzeuge. Hier kannst du auch Startpunkte und Spezialeffekte wie Feuer und Rauch erstellen.
- **TEST:** Diese Registerkarte ist beim Testen deines Spiels hilfreich. Es gibt hier zwei Optionen: *Run* und *Play*. *Run* startet eine Simulation, die zeigt, was mit Blöcken und den anderen Elementen, die sie umgeben, passiert. *Play* ermöglicht es dir, dein Spiel zu spielen.
- **VIEW:** Hier kannst du zwischen den verschiedenen in Roblox Studio verfügbaren Fenstern umschalten. Wenn du ein Fenster benötigst, das geschlossen ist, findest du es auf der Registerkarte **VIEW**.
 - Die beiden Hauptfenster sind **EXPLORER** und **PROPERTIES** (Eigenschaften). Sie werden später in diesem Abschnitt noch ausführlich erläutert.
 - Der Bereich **ACTIONS** enthält verschiedene Einstellungen für die Anzeige. Du kannst hier Schnappschüsse des Bildschirms oder Videos aufnehmen und zwischen Vollbildschirm und Anzeige in einem Fenster umschalten.

- **PLUGINS:** Erweiterungen von Studio. Sie sind standardmäßig nicht installiert und fügen benutzerdefiniertes Verhalten und neue Features hinzu. Du kannst sowohl von der Roblox Community erstellte Plugins installieren als auch deine eigenen Plugins programmieren.

Unterhalb der Menüleiste befindet sich eine Menübandleiste (Abbildung 2.8). Die angezeigten Optionen ändern sich, wenn du die Registerkarte wechselst.

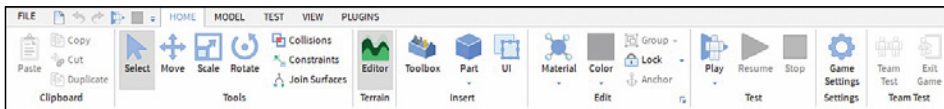


Abb. 2.8: Eine Menübandleiste in Roblox Studio

In den folgenden Abschnitten werden einige der grundlegenden Merkmale und die am häufigsten genutzten Features des Editors erklärt. Außerdem erfährst du, wie du dein Projekt für die Veröffentlichung bei Roblox vorbereitest.

2.3.1 Einrichtung des Arbeitsbereichs im Game-Editor

Da du den Game-Editor zum ersten Mal öffnest, erscheinen auf der linken Seite automatisch zusätzliche Fenster, die du jetzt noch gar nicht benötigst. Um den Arbeitsbereich optimal einzurichten, kannst du die zusätzlichen Fenster schließen, damit du mehr Platz hast.

Standardmäßig werden die Fenster **EXPLORER** und **PROPERTIES** geöffnet und auf der rechten Seite übereinander platziert (Abbildung 2.9).

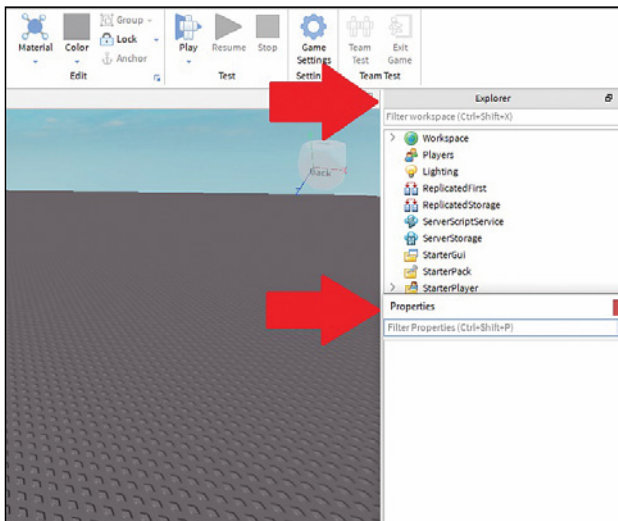


Abb. 2.9: Der Arbeitsbereich mit den übereinander platzierten Fenstern **EXPLORER** und **PROPERTIES**

Features des Game-Editor-Arbeitsbereichs

Wenn du Roblox Studio das nächste Mal startest, ist dein Arbeitsbereich noch immer so, wie du ihn eingerichtet hast. Er bleibt erhalten, es sei denn, du änderst die Anordnung der Fenster. Wenn das Fenster **PROPERTIES** vom Hauptfenster abgelöst wird, kann es schwierig sein, es wieder unterhalb des **EXPLORER**-Fensters anzudocken, es dockt sich nämlich selbst entweder neben oder oberhalb des **EXPLORER**-Fensters an. Um das zu beheben, kannst du beide Fenster schließen. Gehe dann zur Registerkarte **VIEW**, öffne das **EXPLORER**-Fenster, docke es auf der rechten Seite an und schließe es. Mit dem **PROPERTIES**-Fenster verfährt du genauso. Öffne anschließend wieder das **EXPLORER**- und das **PROPERTIES**-Fenster. Jetzt sind die Fenster übereinander ausgerichtet.

2.3.2 Verwendung des Explorer-Fensters

Das **EXPLORER**-Fenster zeigt eine hierarchische Darstellung aller Objekte, die in deinem Spiel verwendet werden. Es ist das wichtigste Fenster, weil es alle organisatorischen, Anzeige- und Test-Features eines Roblox-Spiels enthält.

Das Fenster verwendet das Konzept der Parent-Child-Beziehung¹, um die Objekte zu organisieren. Das Objekt *Game* befindet sich unsichtbar an der Spitze der Hierarchie. In Abbildung 2.10 kannst du beispielsweise sehen, dass unterhalb des Parent-Objekts **WORKSPACE** (Arbeitsbereich) die folgenden Child-Objekte verschachtelt sind: **CAMERA**, **TERRAIN** und **BASEPLATE**.

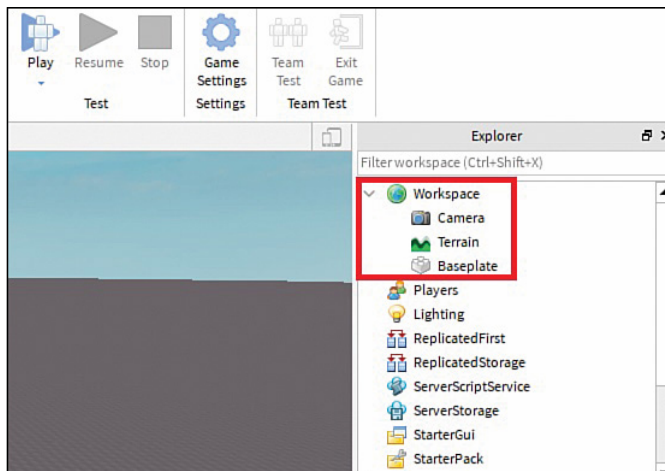


Abb. 2.10: Verschachtelte Objekte unterhalb von **WORKSPACE** im **EXPLORER**-Fenster

- 1 Anm. zur Übersetzung: Hierbei handelt es sich um ein hierarchisches System, das durch eine Baumstruktur abgebildet werden kann. Das Parent-Objekt (Elternteil) ist hierbei einem oder mehreren Child-Objekten (Kindern) übergeordnet.

Wenn du weitere Child-Objekte erstellen möchtest, kannst du den Mauszeiger über **WORKSPACE** platzieren und auf das Plus-Symbol klicken (Abbildung 2.11). Dadurch wird eine Liste aller Objekte angezeigt, die du erstellen kannst. Du kannst ein Objekt auch per Drag & Drop auf das gewünschte Parent-Objekt ziehen.

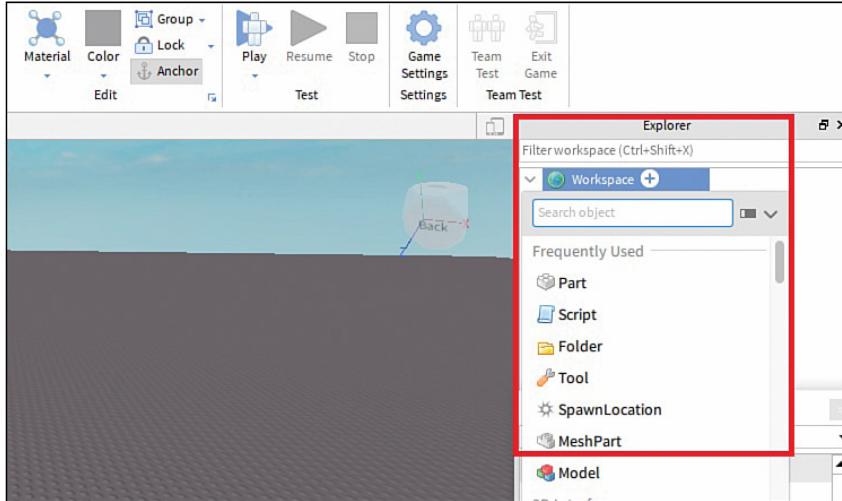


Abb. 2.11: Hinzufügen weiterer Child-Objekte zum Arbeitsbereich

Eins der wichtigsten Child-Objekte, die du verwenden wirst, ist ein *Part*. Dabei handelt es sich um einen grundlegenden Baustein von Roblox. Diese physischen 3-D-Objekte werden auch als *bricks* (Bausteine) bezeichnet. Und wenn sie Teil des Arbeitsbereichs sind, können sie miteinander interagieren.

2.3.3 Erstellen eines Parts

Auf der Registerkarte **HOME** kannst du in der Menübandleiste im Bereich **INSERT** auf **PART** klicken, um einen neuen Part zu erstellen (Abbildung 2.12).

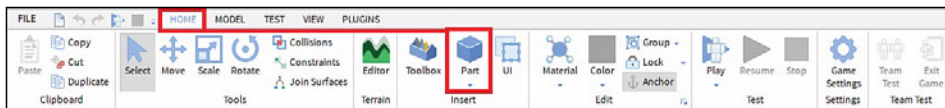


Abb. 2.12: Erstellen eines Parts

Der neue Part erscheint genau in der Mitte deiner Kamera-Ansicht (Abbildung 2.13). Verwende die Kamerasteuerung (Abbildung 2.14), um die Kamera zu verschieben, zu drehen und rein- oder rauszuzoomen.

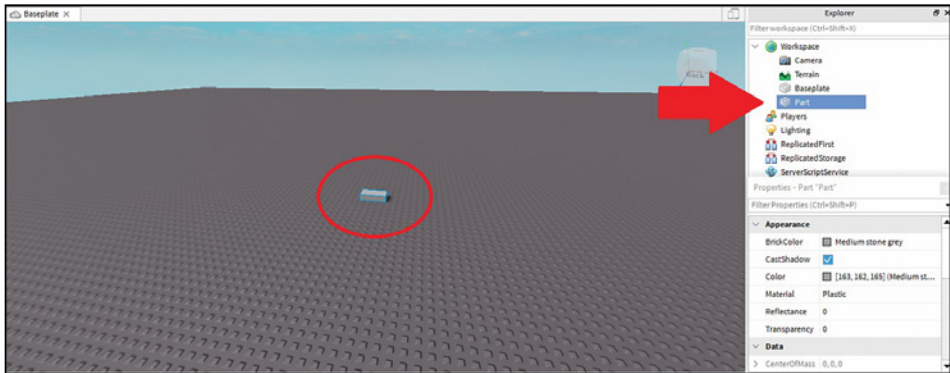


Abb. 2.13: Ein Part erscheint auf der Grundplatte und im Explorer.

Control	Action
W A S D	Move the camera
E	Raise camera up
Q	Lower camera down
Shift	Move camera slower
Right Mouse Button (hold and drag mouse)	Turn camera
Mouse Scroll Wheel	Zoom camera in or out
F	Focus on selected object

Abb. 2.14: Kamerasteuerung

Und so kannst du dem neuen Part einen Namen geben:

1. Doppelklicke auf den Part im EXPLORER-Fenster.
2. Benenne den Part um. In Roblox gibt es die Konvention, für die Namen von Parts den sogenannten *PascalCase* zu verwenden, das heißt, dass der erste Buchstabe jedes Worts großgeschrieben wird, beispielsweise EndZone oder RedBrick.

Namen dürfen Leerzeichen enthalten, aber hier werden keine Leerzeichen verwendet, für den Fall, dass später via Code auf den Part zugegriffen werden soll.

Du kannst den EXPLORER verwenden, um Parts auszuwählen und zu bearbeiten, auch wenn diese im Game-Editor-Fenster gar nicht sichtbar sind.

2.3.4 Verwendung des Properties-Fensters

Wenn du deinem Arbeitsbereich einen Part hinzufügst, werden im PROPERTIES-Fenster neue Informationen angezeigt (Abbildung 2.15).

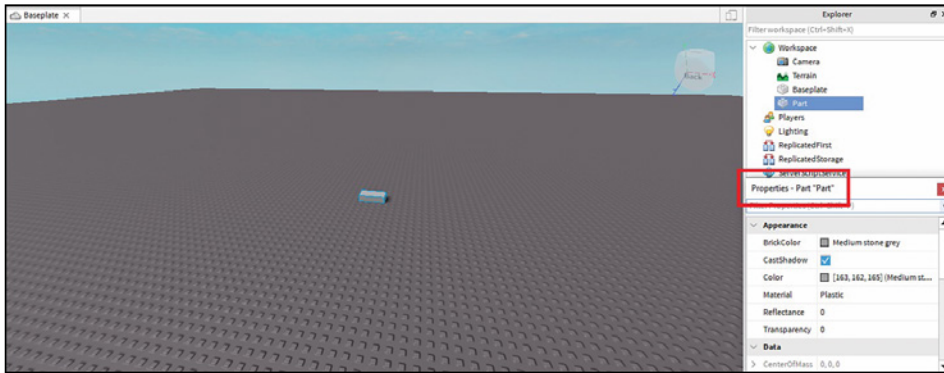


Abb. 2.15: Das Properties-Fenster zeigt die Details des neu hinzugefügten Parts an.

Wie alle Objekte besitzt ein Part Eigenschaften, wie etwa Größe und Farbe. Das PROPERTIES-Fenster zeigt alle Details an, die angeben, wie ein Objekt aussieht und wie es sich verhält. Im nächsten Kapitel werden wir die Eigenschaften eines Parts ausführlicher betrachten und du erfährst, wie du sie bearbeiten kannst.

2.4 Verschiebung, Skalierung und Ausrichtung von Objekten

Du weißt jetzt, wie man einen Part erstellt und nun wirst du ihn in Bewegung versetzen! In Roblox Studio ist es möglich, die Objekte in einer Szene zu verschieben (*Translation*) und zu drehen (*Rotation*). Es gibt verschiedene Möglichkeiten, das gleiche Ergebnis zu erzielen, aber in diesem Abschnitt werden wir ausschließlich die standardmäßig vorhandenen Werkzeuge und Tastenkürzel von Roblox Studio verwenden.

Es gibt zwei Einstellungen, die du verwenden kannst, um beim Verschieben von Parts größeren Einfluss zu nehmen: Einrasten (*Snapping*) und Kollisionen (*Collisions*).

- **Snapping** gibt an, wie weit ein Part jeweils verschoben, skaliert oder gedreht wird. Das erweist sich als nützlich, wenn Objekte erstellt werden, die exakt aneinander ausgerichtet sein sollen wie beispielsweise die Wände eines Gebäudes, die im Winkel von 90 Grad zueinander stehen sollen.
- **Collisions** treten auf, wenn zwei Objekte (oder starre Körper) sich in einem bestimmten Ausmaß überschneiden.

Da diese Einstellungen am häufigsten verwendet werden, wenn man mit zwei oder mehr Parts spielt, solltest du sie zunächst einmal deaktivieren, damit du einen einzelnen Part frei bewegen kannst. Du wirst sie später wieder aktivieren, wenn wir uns damit befassen, wie sie funktionieren.

- Deaktiviere die Kontrollkästchen neben ROTATE oder MOVE auf der Registerkarte MODEL (Abbildung 2.16), um das Snapping (Einrasten) abzuschalten.

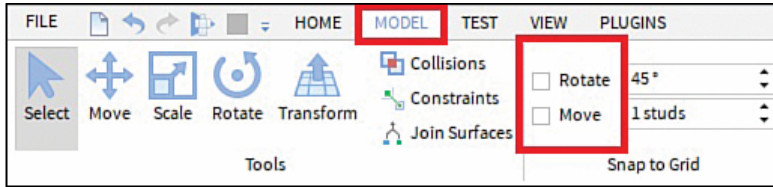


Abb. 2.16: Einrasten deaktivieren

- Kollisionen sind aktiviert, wenn auf der Registerkarte MODEL die entsprechende Schaltfläche grau hervorgehoben ist. Klicke darauf, um Collisions zu deaktivieren (Abbildung 2.17).

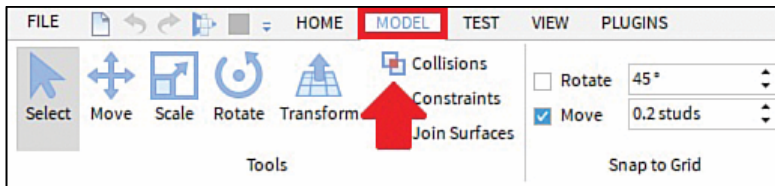


Abb. 2.17: Kollisionen deaktivieren

2.4.1 Verschieben

Jetzt kannst du Objekte nach Belieben verschieben oder umherbewegen. Wähle eine der Registerkarten MODEL oder HOME aus und klicke auf das MOVE-Symbol (Abbildung 2.18).

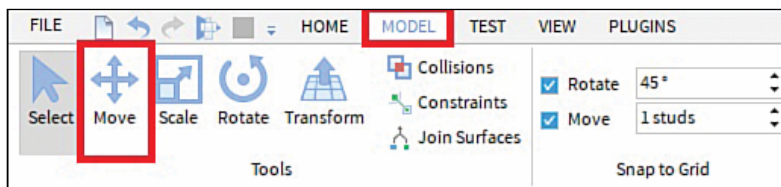


Abb. 2.18: Das Werkzeug zum Bewegen/Verschieben

Nun sollte ein Koordinatensystem für das ausgewählte Objekt erscheinen (Abbildung 2.19). Wenn du einen der Pfeile anklickst und die Maustaste gedrückt hältst, kannst du das Objekt entlang dieser Achse verschieben.

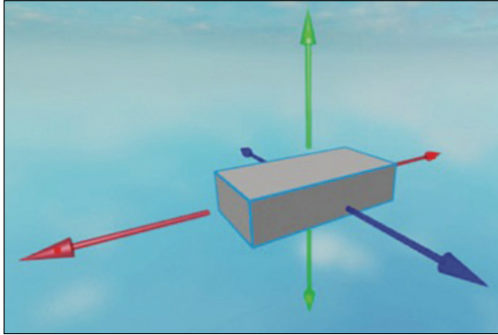


Abb. 2.19: Verschiebung entlang einer Koordinatenachse

2.4.2 Skalieren

Wähle die Registerkarte **MODEL** aus und klicke auf das **SCALE**-Symbol, um Objekte zu skalieren (Abbildung 2.20).

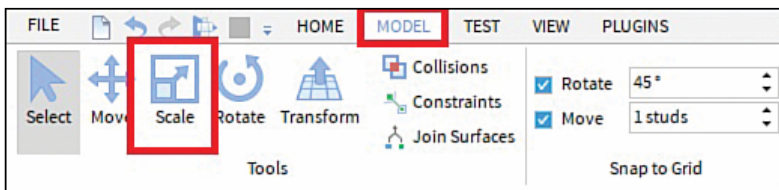


Abb. 2.20: Das Werkzeug zur Skalierung

Die Anzeige des Objekts sollte sich jetzt abermals ändern, allerdings werden jetzt statt der Pfeile Kugeln dargestellt. Wenn du eine der Kugeln anklickst und mit gedrückter Maustaste verschiebst, wird das Objekt entlang dieser Achse skaliert (Abbildung 2.21).

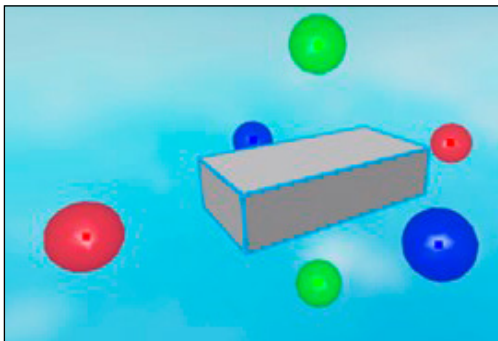


Abb. 2.21: Skalierung

Halte die Taste **Strg** (Windows) oder **⌘** (Mac) gedrückt, um zwei Seiten gleichzeitig mit gedrückter Maustaste zu skalieren.

Halte beim Skalieren die **Shift**-Taste gedrückt, um das aktuelle Seitenverhältnis beizubehalten.

2.4.3 Drehen

Wähle eine der Registerkarten **MODEL** oder **HOME** aus und klicke auf das **ROTATE**-Symbol, um Objekte zu drehen (Abbildung 2.22).

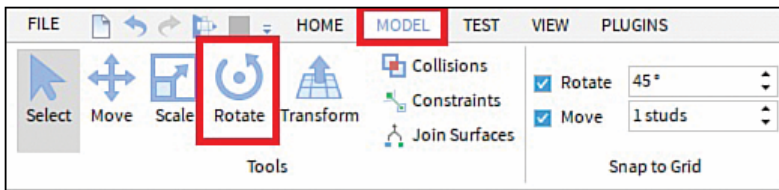


Abb. 2.22: Das Werkzeug zur Drehung

Jetzt sollten beim ausgewählten Objekt durch kreisförmige Linien verbundene Kugeln angezeigt werden (Abbildung 2.23). Wenn du eine der Kugeln anklickst und die Maus bei gedrückter Maustaste bewegst, wird das Objekt um die dazugehörige Achse gedreht.

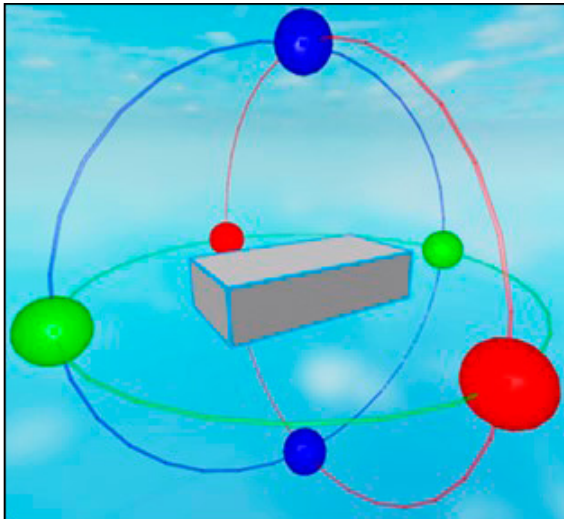


Abb. 2.23: Drehung

2.4.4 Transformieren

Die TRANSFORM-Funktion (Abbildung 2.24) ist von besonderer Bedeutung, da es sich um ein Werkzeug handelt, mit dem du in einem Schritt mehrere Verschiebungen, Skalierungen und Drehungen gleichzeitig durchführen kannst. Stell es dir als eine Verknüpfung von Verschiebung, Skalierung und Drehung vor. Es kann einen Part auf jede erdenkliche Weise verändern. Zudem kann es auf eine Achse beschränkt werden und das Objekt am Gitter ausrichten.

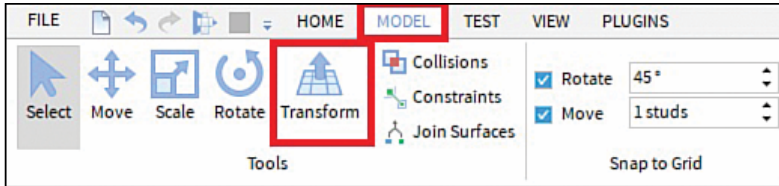


Abb. 2.24: Das TRANSFORM-Werkzeug

Wenn du einen Part auswählst und auf TRANSFORM klickst, werden die Steuerelemente für deinen Part angezeigt (Abbildung 2.25).

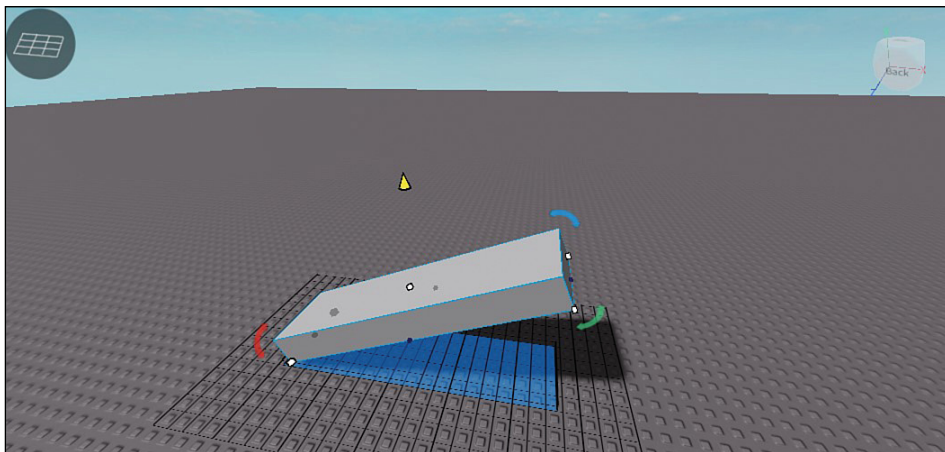


Abb. 2.25: Verwendung des TRANSFORM-Werkzeugs

- Mit dem gelben Kegel kann der Part auf verschiedene Ebenen der Y-Achse verschoben werden. Der Part kann anschließend innerhalb der neuen Ebene verschoben werden.
- Die farbigen Bögen (rot, grün und blau) dienen dazu, den Part um bis zu 360 Grad um die X-, Y- oder Z-Achse zu drehen.

- Die weißen Kästchen werden dazu verwendet, um die Seite des Parts zu skalieren, an der sie sich befinden. Die Skalierung erfolgt in der Einheit *Stud*, die angibt, wie groß die einzelnen Quadrate sind, die die Baseplate bilden.

2.5 Snapping

Jetzt kennst du die Grundlagen des Verschiebens eines einzelnen Parts und wir betrachten Snapping und Collisions erneut. Zur Erinnerung: Snapping gibt an, wie weit ein Part jeweils verschoben, skaliert oder gedreht wird, und ermöglicht es, ein Objekt perfekt auszurichten. Es gibt zwei Snapping-Typen: *Rotation* (Drehung) und *Move* (Verschiebung).

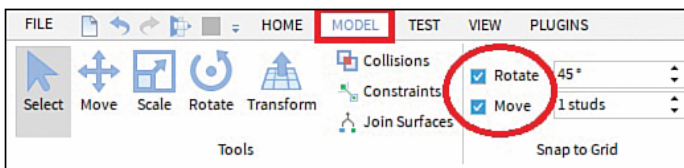


Abb. 2.26: Optionen beim Snapping

- Rotation-Snapping ermöglicht es dir, ein Objekt zu drehen (Angabe in Grad). In diesem Beispiel werden alle Objekte bei jedem Schritt um 45 Grad gedreht.
- Move-Snapping gilt sowohl für Verschiebung als auch für Skalierung. In diesem Beispiel werden alle Objekte bei jedem Schritt um einen Stud verschoben und skaliert.

Wenn du ein Objekt von der Mitte ausgehend skalierst, wird es auf beiden Seiten um einen Stud skaliert. Das entspricht also insgesamt zwei Studs.

Kreuze das Kontrollkästchen neben MOVE oder ROTATE auf der Registerkarte MODEL an, um Snapping wieder zu aktivieren. Anschließend kannst du in den Eingabefeldern für ROTATE oder MOVE die Anzahl der Studs anpassen, um die du Objekte verschieben möchtest (Abbildung 2.26).

2.6 Collisions

Wenn du Collisions wieder aktivierst, kannst du beobachten, wie sie die Bewegungen beeinflussen. In Roblox Studio kannst du mit dem COLLISIONS-Feature steuern, ob Parts einander durchdringen können. Wenn Collisions aktiviert sind, kannst du einen Part nicht an einem Ort platzieren, an dem er sich mit einem anderen Part überschneidet.

Klicke auf der Registerkarte MODEL auf die Schaltfläche COLLISIONS, um Kollisionen einzuschalten. Dadurch werden sie wieder aktiviert und grau hervorgehoben (Abbildung 2.27).

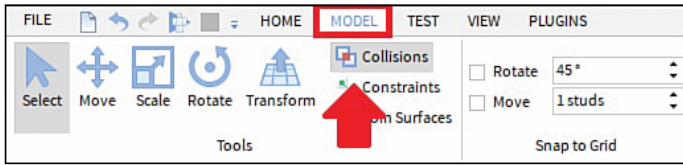


Abb. 2.27: Kollisionen aktivieren

Beim Verschieben von Parts ist dir vielleicht eine weiße Umrisslinie aufgefallen, die immer dann erscheint, wenn ein Part einen anderen berührt. Dadurch wird angezeigt, dass eine Kollision stattfindet. Wir kommen später noch einmal auf Kollisionen zurück.

2.7 Verankern

Wir haben in diesem Kapitel viel zur Erstellung von Parts erörtert, aber was ist zu tun, wenn du möchtest, dass ein Part sich nicht bewegen kann? Wenn ein Part sich nicht bewegen soll, musst du ihn »verankern« (engl. *Anchoring*). Er verbleibt dann stets am selben Ort, auch wenn du das Spiel spielst und andere Spieler und Objekte mit ihm zusammenstoßen. Führe folgende Schritte aus, um einen Part zu verankern:

1. Öffne das PROPERTIES-Fenster.
2. Scrolle nach unten zum Abschnitt BEHAVIOR (Verhalten).
3. Aktiviere das Kontrollkästchen neben ANCHORED (Abbildung 2.28).

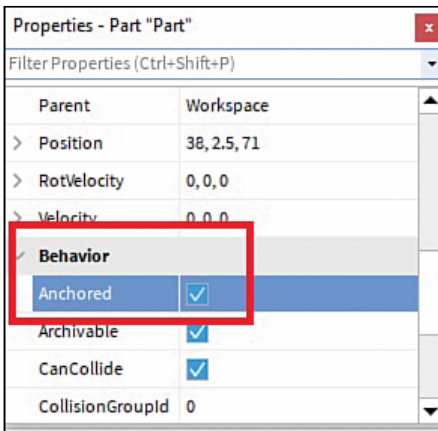


Abb. 2.28: Verankern eines Parts

Du kannst die Verankerung eines Parts auch mit der Schaltfläche ANCHOR auf den Registerkarten MODEL oder HOME aktivieren oder aufheben (Abbildung 2.29).

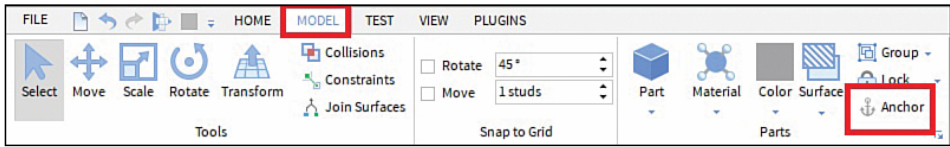


Abb. 2.29: Schaltfläche ANCHOR

Challenge: Parts verankern

Führe folgende Schritte aus, um das Verankern von Parts zu üben:

1. Erstelle einen Part.
2. Verschiebe ihn nach links.
3. Drehe ihn mit SNAP TO GRID um 90 Grad nach links.
4. Überprüfe im PROPERTIES-Fenster, ob er verankert ist.

2.8 Speichern und Veröffentlichung deines Projekts

Du hast im Game-Editor ein Projekt erstellt und möchtest die bei Projekten erzielten Fortschritte natürlich hin und wieder speichern, damit deine Arbeit nicht verloren geht. Und wenn du bereit bist, andere Leute an deinem Werk teilhaben zu lassen, möchtest du dein Projekt sicher auch veröffentlichen.

2.8.1 Speichern deines Projekts

Roblox speichert deine Projekte nicht automatisch, deshalb musst du sie selbst speichern. Es gibt zwei Orte, an denen du Projekte speichern kannst:

- **Auf deinem eigenen Computer:** Klicke in der Menüleiste des Game-Editors in der oberen linken Ecke auf FILE und dann auf SAVE TO FILE. Auf diese Weise bleibt der Template-Name erhalten und das Projekt wird als .rblox-Datei gespeichert. Wenn du stattdessen in diesem Menü die Option SAVE TO FILE AS auswählst, kannst du die Datei umbenennen (Abbildung 2.30).
- **Auf dem Roblox-Server:** Du kannst dein Projekt auch auf dem Roblox-Server speichern, indem du im Menü die Option SAVE TO ROBLOX AS verwendest. Deine Arbeit wird an einem sicheren Ort auf dem Roblox-Server gespeichert, ist der Öffentlichkeit aber nicht zugänglich.

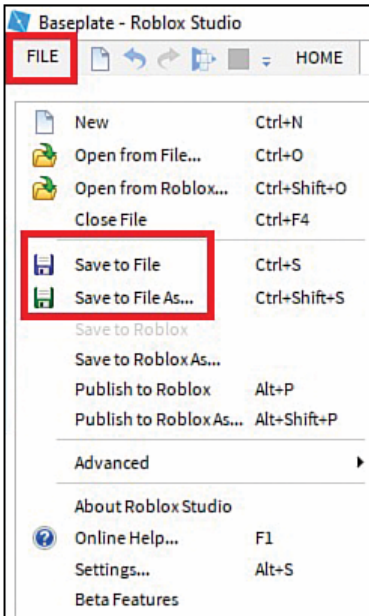


Abb. 2.30: Die Befehle SAVE TO FILE und SAVE TO FILE AS befinden sich im FILE-Menü.

2.8.2 Veröffentlichung deines Projekts

Was nützt es, ein Spiel zu erstellen, wenn niemand es spielen kann? Um es öffentlich zugänglich zu machen und Geld damit zu verdienen, muss das Projekt durch Auswahl der Option PUBLISH TO ROBLOX veröffentlicht werden. Die Veröffentlichung ermöglicht es anderen Spielern, dein Spiel bei Roblox zu spielen. Führe folgende Schritte aus, um dein Spiel bei Roblox zu veröffentlichen:

1. Wähle den Menüpunkt FILE|PUBLISH TO ROBLOX aus, um das entsprechende Fenster zu öffnen.
2. Gib einen Namen und optional eine Beschreibung ein.
3. Klicke auf die Schaltfläche CREATE.

2.8.3 Projekt wieder öffnen

Wenn du das Projekt, an dem du gearbeitet hast, wieder öffnen möchtest, kannst du es auf der Startseite von Studio (Abbildung 2.31) folgendermaßen finden:

1. **File-Menü:** Wähle den Menüpunkt FILE|OPEN.
2. **My Games:** Wenn du dein Spiel bei Roblox veröffentlicht hast, findest du dein Spiel unter MY GAMES.
3. **Recent:** Unter RECENT findest du alle Dateien, die du in letzter Zeit geöffnet hast.

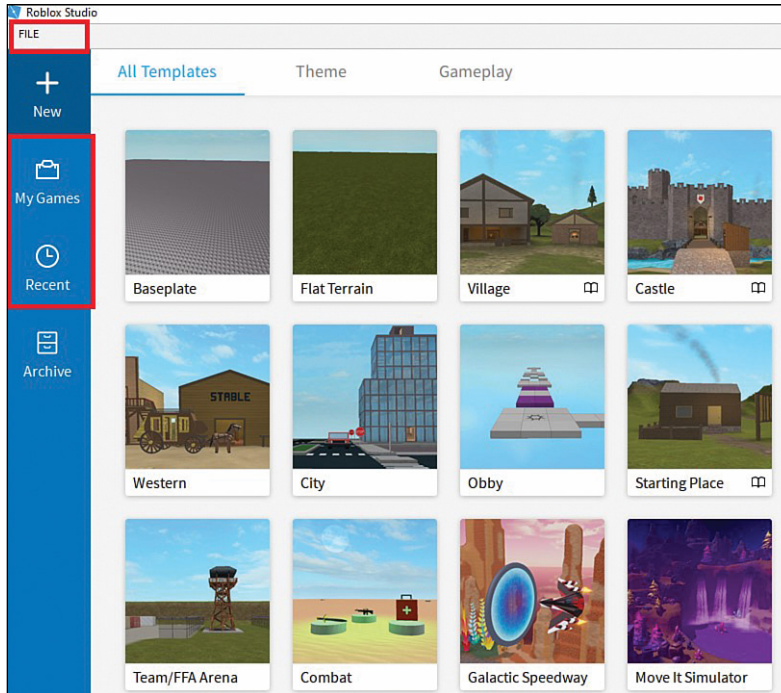


Abb. 2.31: Erneutes Öffnen vorheriger Projekte auf der Startseite von Studio

2.9 Spieltest

Ein Spieltest ist das Spielen eines Spiels, um sicherzustellen, dass alles funktioniert, und um herauszufinden, ob es nicht sogar noch verbessert werden kann. Diesen Schritt solltest du nicht auslassen, weil er für ein erfolgreiches Spiel entscheidend ist. Es ist nur vernünftig, bei jeder Änderung am Spiel einen Spieltest durchzuführen. Du solltest dein Spiel außerdem in verschiedenen Modi testen. Du kannst im Spiel-Modus zwar Änderungen vornehmen, diese werden jedoch nicht gespeichert. Du musst sie erneut vornehmen, wenn du wieder den Bearbeiten-Modus verwendest.

Spieltest in der Praxis

- Vergewissere dich, dass dein Spiel funktioniert und insbesondere die gerade vorgenommenen Änderungen.
- Halte nach Bereichen Ausschau, die verbessert werden können.
- Wenn du Templates erkundest oder einen Spieltest damit durchführst, solltest du dich vergewissern, dass du dir genau angesehen hast, wie die Parts heißen und wie sie gruppiert sind.

2.9.1 Spieltest durchführen

Führe beim Spieltest die folgenden Schritte aus:

1. Speichere dein Spiel. Vergiss nicht, den Dateinamen zu ändern.
2. Klicke auf die Schaltfläche **PLAY** in der oberen Menüleiste. Auf der Registerkarte **HOME** befindet sich im Menü **TEST** ebenfalls eine Schaltfläche **PLAY** (Abbildung 2.32).

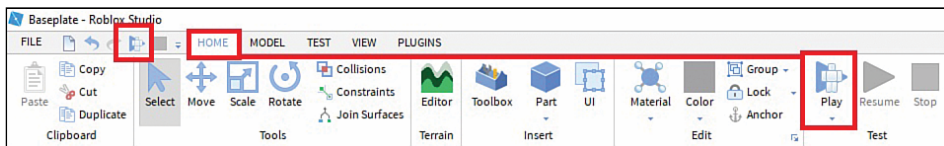


Abb. 2.32: Die Schaltfläche **PLAY** für den Spieltest

2.9.2 Spieltest beenden

Klicke auf die rote Schaltfläche **STOP** in der oberen Menüleiste oder im Menü **TEST** (Abbildung 2.33), um den Spieltest zu beenden. Beende den Spieltest, bevor du Änderungen vornimmst, denn im Spiel-Modus vorgenommene Änderungen können nicht gespeichert werden.

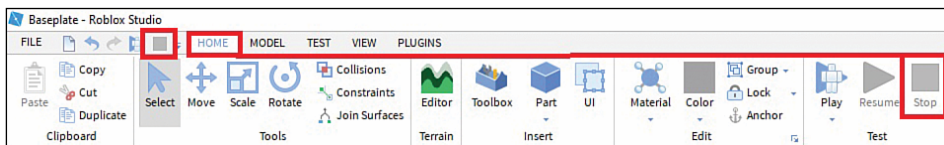


Abb. 2.33: Die Schaltfläche **STOP** zum Beenden des Spieltests

Challenge: Spieltest-Übungen

Führe Spieltests mit den folgenden beiden Templates durch:

- Village
- Obby

Vor dem Spieltest kannst du die Orte ändern, an denen die Parts platziert sind. Du kannst Parts mit gedrückter Maustaste verschieben und im **PROPERTIES**-Fenster beobachten, wie sich ihre Eigenschaften verändern, oder auch das Material ändern oder sie löschen. Vergiss nicht, das Template unter einem neuen Namen zu speichern oder zu veröffentlichen. Und wenn du versuchst, neue Parts oder Effekte hinzuzufügen, musst du dich vergewissern, dass sie sich nicht im Spieltest-Modus befinden.

2.10 Zusammenfassung

In diesem Kapitel hast du erfahren, wie einfach es ist, Spiele mit Roblox Studio zu erstellen und mit Millionen Spielern zu teilen. Du hast gelernt, wie man Roblox Studio installiert und verwendet und wie man den Arbeitsbereich einrichtet, Änderungen an Templates vornimmt und Spiele bei Roblox speichert und veröffentlicht, um sie der Öffentlichkeit zugänglich zu machen. Außerdem hast du gelernt, wie du Spieltests deiner Änderungen durchführst, um den Erfolg deines Spiels zu gewährleisten.

2.10.1 Fragen und Antworten

Was kann ich tun, wenn Studio sich nicht installieren lässt?

Vergewissere dich, dass die minimalen Systemanforderungen erfüllt sind. Wenn das nicht der Fall ist, lässt sich Studio aber dennoch installieren, kann es Probleme bei der Ausführung des Programms geben.

Kann ich ein Template modifizieren?

Templates sind vorgefertigte Projekte, die du als Ausgangspunkt für deine eigenen Spiele verwenden kannst.

Kann ich während eines Spieltests vorgenommene Änderungen speichern?

Im Spiel-Modus vorgenommene Änderungen werden nicht gespeichert. Du musst sie im Bearbeiten-Modus erneut vornehmen.

2.10.2 Workshop

Jetzt soll das Gelernte kurz wiederholt werden. Nimm dir einen Moment Zeit, um die folgenden Fragen zu beantworten.

Quiz

1. Wie organisierst du deinen Arbeitsbereich?
2. Welche gängigen Templates können als Ausgangspunkt für die Entwicklung von Grund auf verwendet werden?
3. Wie kannst du deinen Avatar während eines Spieltests bewegen?
4. Richtig oder falsch: Die Veröffentlichung deines Projekts bei Roblox macht es jedem zugänglich.
5. Richtig oder falsch: Mit dem Transform-Werkzeug können mehrere Schritte gleichzeitig durchgeführt werden.

Lösungen

1. Das Schließen zusätzlicher Fenster verschafft dir mehr Platz, um zu sehen, was du gerade machst und sorgt dafür, dass EXPLORER- und PROPERTIES-Fenster vertikal aneinander ausgerichtet sind.
2. Baseplate und Flat Terrain sind zwei häufig verwendete Templates, die ein Spieleentwickler als Ausgangspunkt für eine komplette Spielwelt nutzen kann.
3. Du kannst die Tasten **W** **A** **S** **D** oder die Pfeiltasten verwenden, um dich zu bewegen.
4. Richtig. Bei der Veröffentlichung wird deine Arbeit an einem sicheren Ort gespeichert, was es anderen Spielern bei Roblox ermöglicht, dein Spiel zu spielen. (Nach der Erstveröffentlichung kannst du unter GAME SETTINGS (Spieleinstellungen) festlegen, dass dein Spiel allen zugänglich ist.)
5. Richtig. TRANSFORM ist ein Werkzeug, mit dem du in einem Schritt mehrere Verschiebungen, Skalierungen und Drehungen gleichzeitig durchführen kannst

2.10.3 Aufgaben

Führe die folgenden Aufgaben aus, um weitere Einblicke in Roblox Studio zu gewinnen.

1. Öffne ein neues Baseplate-Template.
2. Füge auf der Registerkarte HOME einen neuen Part hinzu.
3. Suche im EXPLORER-Fenster unter WORKSPACE nach dem neu hinzugefügten Part und benenne ihn in CENTERPART um.

4. Benenne die Baseplate um, speichere sie und veröffentliche sie bei Roblox.
5. Führe einen Spieltest durch.

In der zweiten Aufgabe werden mehrere Dinge, die du in diesem und dem vorigen Kapitel gelernt hast, miteinander kombiniert. Wenn du nicht weiterkommst, kannst du auf den vorangehenden Seiten in diesem Kapitel nachschlagen. Du wirst einen einfachen Hindernisparkour erstellen, der bei Roblox üblicherweise als »Obby« (von engl. *obstacle course*) bezeichnet wird.

1. Verwende zunächst einmal einige Parts. Vergewissere dich, dass ANCHORED aktiviert ist, und platziere sie am Himmel.
2. Füge an einem Ende der Parts einen weiteren Part hinzu. Er wird der Anfang der Hindernisbahn deines Spiels sein. Vergewissere dich, dass ANCHORED hier ebenfalls aktiviert ist.
3. Füge einen letzten Part am anderen Ende der Parts hinzu. Er wird das Ende der Hindernisbahn deines Spiels sein.
4. Führe einen Spieltest durch. Probiere es aus, über den Startpunkt zu fliegen, indem du auf der Registerkarte HOME auf den blauen Pfeil unter PLAY klickst und PLAY HERE auswählst.
5. **Zusatz:** Füge aus dem Bereich GAMEPLAY auf der Registerkarte MODEL ein SPAWN-Objekt (das markiert einen Startpunkt) hinzu, um zu vermeiden, PLAY HERE betätigen zu müssen, und dass alle Spieler am Anfang starten. (ANCHORED ist hierfür standardmäßig aktiviert.)

Tipps

- Füge mindestens fünf oder sechs Parts unterschiedlicher Größe und Form hinzu, um ein Springspiel für die Spieler zu erstellen. Der erste Sprung sollte leichter sein als nachfolgende Sprünge.
- Führe während des gesamten Entstehungsprozesses immer wieder Spieltests durch, um zu gewährleisten, dass alle Sprünge machbar und alle Parts fest verankert sind.

Physik-Engine

Die Themen in diesem Kapitel:

- Attachments (Verbindungspunkte) und Constraints (Verbindungen)
- Die `CANCOLLIDE`-Eigenschaft
- Hinges (Scharniere) und Springs (Federn)
- Wie man Motoren verwendet

Nachdem du erfahren hast, wie man Parts in Roblox bearbeiten kann, wirst du in diesem Kapitel lernen, wie die Physik-Engine verwendet wird, um realistische interaktive Umgebungen zu erstellen. Wenn du eine funktionierende Tür oder einen sich drehenden Ventilator benötigst, musst du auf die Physik-Engine zurückgreifen.

Die Physik-Engine legt fest, wie sich ein Part, sei es ein Ziegelstein, ein Keil oder ein Zylinder, in einem Roblox-Spiel bewegt. Die Engine ahmt die Physik in der echten Welt nach und ermöglicht es so, auf einfache Art und Weise bewegliche Vorrichtungen zu entwickeln, beispielsweise funktionierende Schränke oder komplexe Apparaturen. Mit der Engine richtest du Motoren, Scharniere und Federn ein und bearbeitest ihre Eigenschaften, um festzulegen, wie schnell oder langsam sie im Spiel funktionieren.

Im folgenden Abschnitt betrachten wir zwei verschiedene Möglichkeiten, eine Tür zu erstellen. Die erste Methode zeigt dir, wie eine einfache Tür erstellt wird, durch die Spieler gehen können, allerdings ohne die physikalischen Gesetze zu berücksichtigen. Die zweite Methode hingegen macht sich die Features der Physik-Engine zunutze, um eine Tür zu erstellen, die sich öffnet und schließt, wenn Spieler sie passieren.

In diesem Kapitel verwendest du Studio, um

- eine Tür zu erstellen,
- einen Spieler durch die Tür zu bewegen, indem `CANCOLLIDE` deaktiviert wird,
- `CANCOLLIDE` wieder zu aktivieren, um eine realistischere Tür bereitzustellen, die Scharniere und Federn hat.

4.1 Verwendung von Attachments und Constraints

Vor dem Erstellen einer Tür musst du zwei Schlüsselemente mechanischer Konstruktionen verstehen: Attachments und Constraints. Das Attachment ist die Stelle, an der ein Objekt mit einem Part verbunden ist. Alle Attachments haben einen Part als Parent (Elternteil) (Abbildung 4.1).

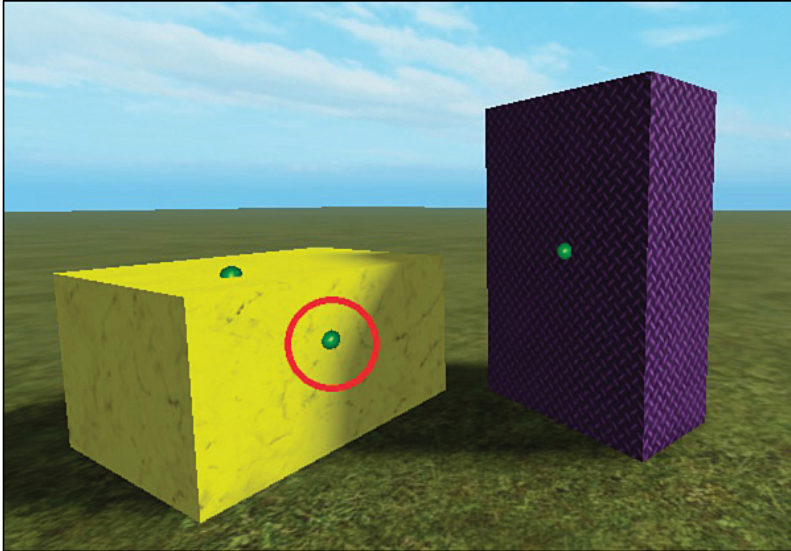


Abb. 4.1: Attachment-Punkte eines Parts

Ein Constraint verbindet zwei Attachments miteinander. Constraints sind Bauelemente wie Stangen, Motoren, Scharniere, Federn und andere, die dazu verwendet werden können, mechanische Konstruktionen zu bauen. Abbildung 4.2 zeigt ein Beispiel, in dem eine Stange als Constraint verwendet wird.

In diesem Beispiel verwendest du eine Stange als Constraint, um einen nicht verankerten Part mit einem verankerten zu verbinden:

1. Erstelle zwei schwebende Parts, die sich übereinander befinden (Abbildung 4.3). Verankere den oberen Part, den unteren jedoch nicht.
2. Aktiviere CONSTRAINT DETAILS auf der Registerkarte MODEL (Abbildung 4.4). So kannst du die erstellten Constraints und Attachments besser sehen.

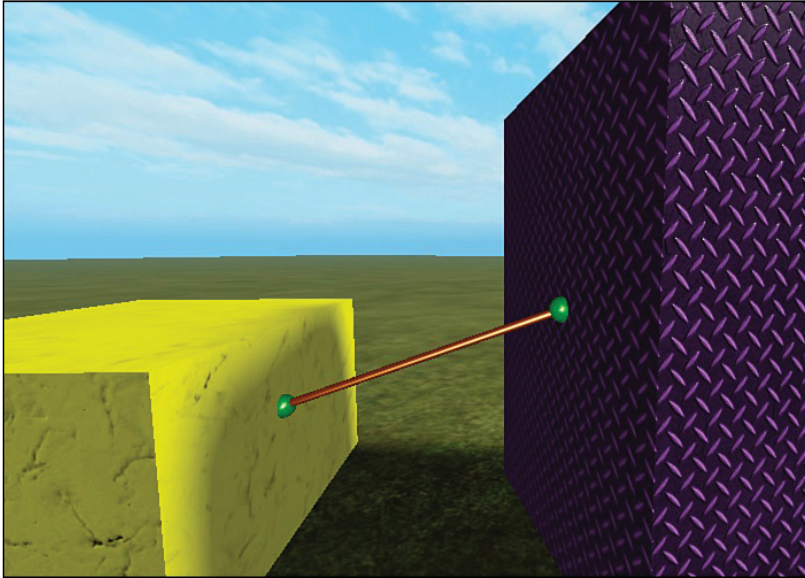


Abb. 4.2: Eine Stange als Constraint

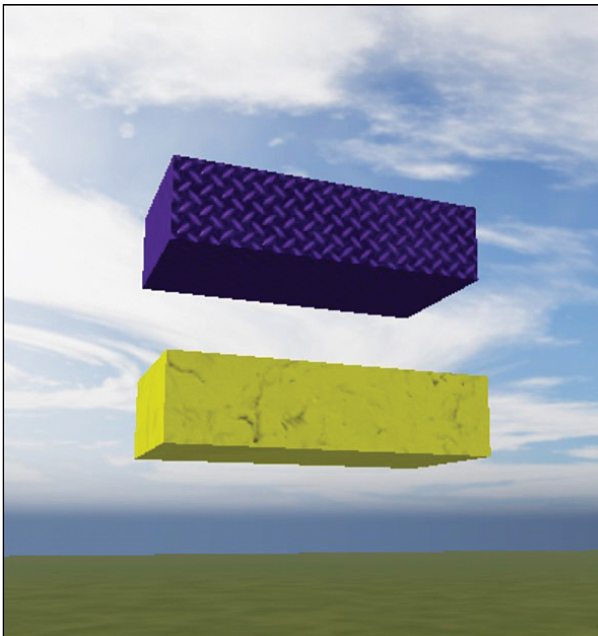


Abb. 4.3: Zwei Parts, die durch eine Stange verbunden werden

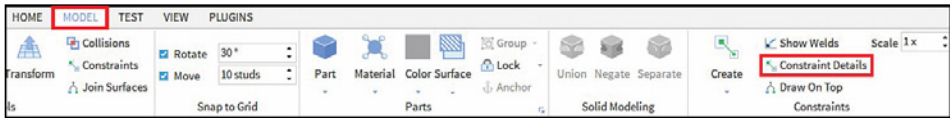


Abb. 4.4: Schaltfläche CONSTRAINT DETAILS

3. Klicke auf der Registerkarte MODEL auf die Schaltfläche CREATE und wähle im Menü ROD CONSTRAINT aus.
4. Klicke auf der Unterseite des oberen Ziegelsteins auf den Punkt, mit dem die Stange verbunden werden soll, und klicke anschließend auf die Oberseite des unteren Ziegelsteins. Die Stange wird zwischen zwei grünen Attachments eingefügt (Abbildung 4.5).

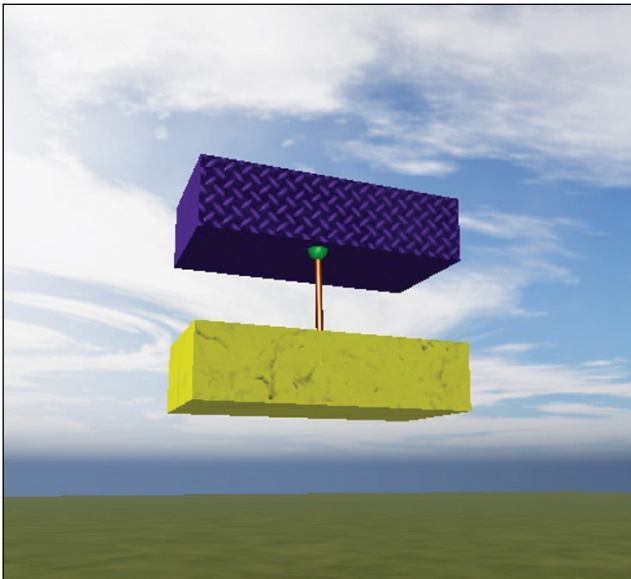


Abb. 4.5: Zwei Parts sind durch einen Constraint miteinander verbunden.

5. Führe einen Spieltest durch. Du wirst feststellen, dass der nicht verankerte Part am verankerten Part befestigt ist.

4.2 Erstellen einer Tür

Jetzt hast du etwas Übung im Umgang mit Attachments und Constraints und kannst zwei Möglichkeiten ausprobieren, eine Tür zu erstellen. Du musst bei beiden Methoden eine einfache Tür aus mehreren Parts aufbauen, indem du das einsetzt, was du im letzten Kapitel gelernt hast.

1. Verwende drei Parts, um den Türrahmen zu erstellen, und einen weiteren Part für die Tür selbst. Verwende SNAP TO GRID, um die Objekte aneinander auszurichten.
2. Verankere den Türrahmen, aber nicht die Tür selbst.
3. Füge einen Türgriff hinzu, damit du erkennen kannst, welche Seite der Tür du vor dir hast (Abbildung 4.6).

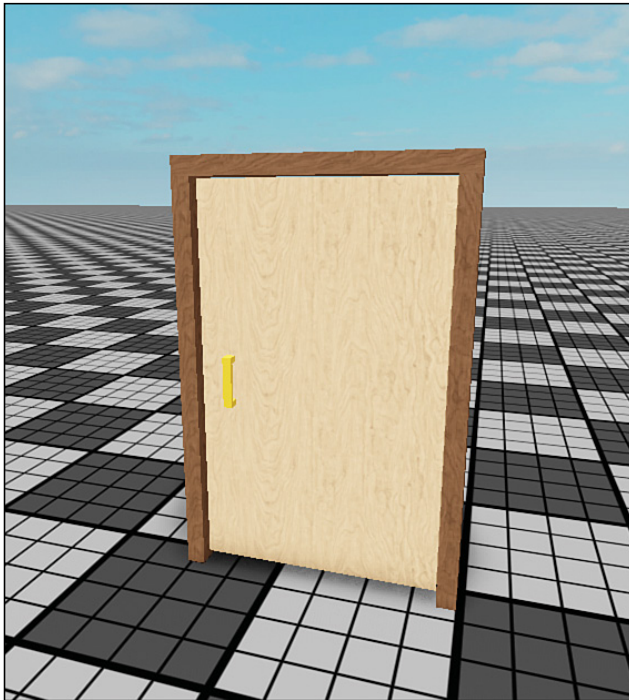


Abb. 4.6: Eine Tür mit Türgriff

Damit der Türgriff an der Tür befestigt bleibt, wenn sie bewegt wird, benötigst du einen *Weld* (Verschweißung). Das ist ein Constraint, das dazu verwendet wird, Objekte fest aneinander zu befestigen. Führe die folgenden Schritte aus, um einen Weld hinzuzufügen:

1. Klicke auf **CREATE** und wähle im Menü **WELD** aus (Abbildung 4.7).
2. Klicke auf die beiden Parts, die du miteinander »verschweißen« möchtest. Mit einem Constraint kannst du nur zwei Objekte miteinander verbinden.
3. Fahre damit fort, die anderen Parts miteinander zu verschweißen, bis die Aufgabe erfolgreich erledigt ist.

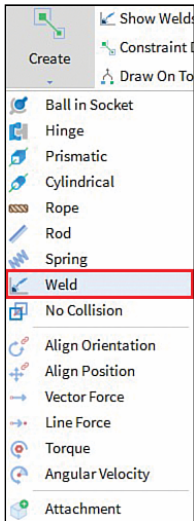


Abb. 4.7: Auswahl der Option `WELD`

Abbildung 4.8 zeigt, welche Parts verschweißt (`WELDED`) sind und welche Parts der Tür verankert (`ANCHORED`) sind, damit gewährleistet ist, dass sie den Gesetzen der Physik gehorcht. Wie du an diesem Beispiel siehst, ist nicht nur der Türgriff mit der Tür verbunden, auch die Tür selbst besteht aus mehreren Parts, die miteinander verschweißt sind.

Jetzt hast du ein Objekt, das aussieht wie eine Tür, es muss aber auch wie eine Tür funktionieren und es einem Spieler ermöglichen, die Tür zu durchschreiten.



Abb. 4.8: Verschweißte und verankerte Parts

Stichwortverzeichnis

Symbole

24-Stunden-Format 313
.Touched 249

A

Adornee 272
Analytics 468
Anchored 416
AnchorPoint 269
AngularVelocity 100
Animation 287
 Einstellungen 327
 Event 329
 Priorität 328
 speichern 322
Anweisung
 bedingte 229
Appearance 341
Argument 226
Array 230
Asset Manager 192
Atmosphäre 147
Attachment 84
Audiodatei 305
 Lupe 189
Audioformate 196
Aufkleber 72

B

Baseplate 45
Beam 165, 170
Bedingte Anweisung 229
Befehlsleiste 436

Beleuchtung 131, 133
 Ambient 135
 Appearance 133
 Behavior 133
 Brightness 135
 ClockTime 137
 Data 133, 136
 EnvironmentDiffuseScale 135
 EnvironmentSpecularScale 135
 Exposure 133
 GlobalShadows 135
 TimeOfDay 137
Beleuchtungseffekt 137
Benachrichtigung 466
Benutzeranzeige 464
Bildformate 194
BillboardGui 266
BindableEvent 235
BindAction 421
BindToRenderStep() 403
Biom 106
Bloom-Effekt 138
BlurEffect 139
Body Part 325
Bogenmaß 291
Breakpoint 237

C

Callback 447
CamelCase 226, 260
CameraOffset 403
CanCollide 89, 416
Cassel, Erik 17
Catalog 27

CCPA 434
 CelestialBodiesShown 160
 CFrame 288
 Koordinaten 350
 CFrame.Angles 291
 Chang, Henry 20
 Chat 209
 China 434
 ClickDetector 289
 Click-Through-Rate 464
 Client-Server-Modell 367
 Coding Workspace 220
 Collision 53, 58
 CollisionFidelity 243, 416
 Collision Groups Editor 246
 Color 152
 ColorCorrection 139
 Color Map 126
 ColorShift_Top 160
 Command Bar 436
 Connect 249
 Constraint 84
 Consumables 445
 ContextActionService 421
 Copy-Werkzeug 123
 Countdown 282
 Curtis, Matt 420
 CurveSize 172

D

Dämpfung 97
 Damping 97
 Data StoreService 356
 Daten
 personenbezogene 435
 Datenspeicher
 Datentypen 356
 dauerhafter 356
 Datentyp 256
 Datenverlust 363
 debounce 250
 Debugger 237

Debugging 237
 Decal 72
 erstellen 73
 importieren 196
 Decay 154
 Delete-Werkzeug 123
 deltaTime 408
 Density 149
 DepthOfField 140
 Developer Exchange 450
 Dictionary 230
 Docking, Theo 19
 Drehen 56
 Drehmoment 100
 DSGVO 434
 Dunst 151

E

Easing
 lineares 323
 EasingDirection 298
 EasingStyle 297
 Eigenschaft 29
 ändern 222
 else-Block 229
 elseif-Block 230
 EmissionDirection 166
 Emulation Device Manager 422
 end 226
 Enum 256
 Enumeration 256
 Enum.ProductPurchaseDecision.NotProces-
 sedYet 447
 Ereignis 228
 Erode-Werkzeug 113
 Erscheinungsbild
 Part 67
 Event 228
 benutzerdefiniertes 235
 hinzufügen/verschieben/löschen/
 klonen 330
 Implementierung in Scripts 330

EventPublisher 236
 EventSubscriber 236
 Explorer-Fenster 50
 Explosion 227

F

Farbkorrektur 139
 Farbsättigung 139
 Farbwahldialog 69
 Feder 90, 94
 Fehlerhandhabung 362
 Feuerstelle 178
 Fill-Werkzeug 123
 Filtering Enabled 32
 Fixed Plane 114
 Flatten Mode 114
 Flatten-Werkzeug 114
 Flat Terrain 45
 for-Schleife 233
 Frame 275, 403
 Full Body 325
 function 226
 Funktion 226
 erstellen 226
 globale 235

G

Game-Editor 47
 Game-Loop 389, 391
 Game Pass 441
 GDPR 434
 Geisterbild 151
 Gelände 105
 Generate-Werkzeug 106
 GetAsync() 360
 GetKeyframeMarkerReached() 329
 GetNetworkOwner() 378
 GetPolicyInfoForPlayerAsync 433
 GetPropertyChangedSignal 256, 419
 GetService() 351
 GetTeams() 377
 Gewalt 434

Ghosting 151
 Glare 153
 Gleichheitszeichen-Operator (=) 223
 Glücksspiel 434
 Griffstellung 341
 Grip 341
 Group 201
 Grow-Werkzeug 112
 Gruppeneigentümer 205
 Gruppenmitglied 205
 Gruppenspiel 204
 GUI-Element 274
 GUI (Graphical User Interface) 265
 Gültigkeitsbereich 223, 234

H

Handle 339
 Haze 151
 Height Map 124
 Himmelskörper 159
 HingeConstraint 90
 Humanoid 253
 HumanoidRootPart 254, 348

I

Icon 457
 Identität 26
 if-Block 229
 ImageButton 274
 ImageLabel 274
 IncrementAsync() 361
 Inkrementwert 233
 Instance.new() 259
 InvokeClient() 370
 InvokeServer() 370
 iPairs() 234
 isFriendsWith() 377

K

Kamera
 Abstand 403
 Kameraeigenschaften 399

Karteneinstellungen 107
 Keyframe 316
 hinzufügen/löschen/klonen/
 bewegen 320
 Kinematik
 inverse 324
 Klanglandschaft 313
 Klassen-API 229
 Klick 464
 Kollision 54, 58
 Kollisionserkennung 249
 Kollisionsgruppe 245
 Kommentar
 hinzufügen 224
 Konsole 414
 Kontrast 139
 Krümmung 172

L

Landschaft 105
 Lautstärke 308
 Layout 278
 Layoutreihenfolge 278
 Leuchteffekt 273
 LightInfluence 273
 Lighting 133
 Lighting-Objekt
 Eigenschaften 140
 local 226, 234
 Log-Datei 238
 Lokalisierung 429
 Looping 327
 Lootbox 434
 LOUD_SNAKE_CASE 260
 Lua 219

M

Map Settings 107
 Marketplace Fee 448
 MarketPlaceService 443
 Material 69
 Materialeinstellungen 107

Material Settings 107, 116
 Merge Empty 120
 Mesh 191, 415
 Größe 191
 Wiederverwendung 415
 MeshPart 189
 Metaverse 18
 Mobilgerät 420
 Modell
 bewegen 298
 einfügen 187
 erstellen 184
 hochladen 185
 Zugriff 187
 Moderation 126, 191
 Module-Script 383
 Aufbau 384
 MoonAngularSize 159
 MoonTextureId 159
 Motor 98
 MotorMaxTorque 100
 MouseEnter 276
 MouseLeave 276
 Move-Werkzeug 120

N

Namenskonvention 52
 Network-Ownership 378
 Network-Simulator 346
 nil 376
 NPC 253

O

Obby 44, 66
 Objekt
 bewegen 289
 drehen 56
 Elternteil 418
 skalieren 55
 transformieren 57
 verschieben 54
 Offset 78, 150

Operator 474
 Optimierung 414
 Owner 378

P

Package 210
 aktualisieren 213
 löschen 212
 Package-Toolbox 212
 Paint-Werkzeug 115
 pairs() 234
 Paket *siehe Package*
 Part 29, 51
 anheften 326
 Erscheinungsbild 67
 erstellen 51
 ParticleEmitter
 Eigenschaften 169
 Partikel 165
 anpassen 167
 Farbe ändern 168
 Partikelerzeuger 166
 PascalCase 52, 260
 Paste-Werkzeug 123
 pcall 362
 Performance
 Verbesserung 413
 PhysicsService 247
 Physik-Engine 33, 83, 416
 Place
 hinzufügen 202
 Plattform 35
 PlayerGui 266
 PlayerMembershipChanged 449
 Plugin 31
 PointLight 141
 Portal 348
 Pose 316
 PreciseConvexDecomposition 245
 PrimaryPart 184
 print-Funktion 221
 processReceipt 447

Projekt
 öffnen 61
 speichern 60
 veröffentlichen 61
 PromptGamePassPurchase 443
 PromptProductPurchase 446
 Properties-Fenster 52
 Protokoll 238
 Provision 448

R

random() 301
 Reflexionsgrad 71
 Regelkonformität 433
 Region 118
 RemoteEvent 369
 RemoteFunction 369
 RemoveAsync() 361
 RenderFidelity 415
 Render-Step 402
 RenderStepped-Event 406
 repeat-until-Schleife 232
 require() 385
 Resize-Werkzeug 122
 Rig 316
 Roblox Developers Conference 24
 Roblox Payouts 448
 Roblox Premium 448
 Robux 21
 Rolle
 Konfiguration 204
 Zuweisung 205
 RollOffMode 308
 Rotate 56
 Rotation 291
 RunService 403
 IsClient 396
 IsServer 396

S

Sarwar, Ashan 19
 Scharnier 90

Schatten 417
 Schleife 231
 Schlüsselbild 316
 Scope 223, 234
 ScreenGui 266
 Script 220
 erstellen 220
 Script-Editor 220
 Sea-Level-Werkzeug 117
 Segments 174
 Seitenabruf 464
 Seitenverhältnis
 beibehalten 56
 ServerScriptService 220
 SetAsync() 361
 SetPrimaryPartCFrame() 299
 Simulator 422
 Skalieren 55
 Skalierung 268
 Skybox 147, 156
 erstellen 156
 Smooth-Werkzeug 113
 Snapping 53, 58
 Sortierreihenfolge 278
 Sound
 gruppieren 309
 SoundGroup 310
 Sound-Objekt 307
 Sounds
 importieren 196
 Soundtrack 303
 Speicherbedarf 413
 Speichern 60
 Spielerdaten
 löschen 435
 Spielkonsole 424
 Spielrunden 391
 Spieltest 62
 Spieluniversum 350
 Sponsoranzeige 461
 SpotLight 141
 SpringConstraint 94

Standardanimation
 ersetzen 332
 Standardmodell 210
 StarCount 159
 StarterPack 339
 Statistik 468
 Steifheit 97
 Stiffness 97
 Strahleneffekt 176
 Streaming 416
 Stud 58
 Studio
 Installation 42
 Systemanforderungen 42
 Subtract-Werkzeug 111
 SunAngularSize 159
 SunRays-Effekt 137
 SunTextureId 159
 SurfaceGui 266, 270
 SurfaceLight 142
 Sutrave, Swathi 20

T

Tabelle 230
 Team
 hinzufügen 375
 Zuordnung von Spielern 376
 Team Create 201
 aktivieren 205
 Benutzer hinzufügen 206
 deaktivieren 209
 TeamService 377
 Teleportation 347
 Client 352
 Server 353
 TeleportService 351
 Template 44
 Terrain 30
 Terrain-Editor 105
 Test
 Spieltest 62

Text
 anzeigen 268
 TextButton 274
 Textfassung 430
 Textlabel 274
 Textur 72, 76
 importieren 194
 Wiederverwendung 416
 Theme 45
 Timer 283
 Tool 337
 Eigenschaften 346
 erstellen 339
 Touched 249
 Touched-Event 228
 Trailer 458
 Transformieren 57
 Transparenz 71
 Tür
 erstellen 86
 mit Scharnieren 90
 tween
 Play() 302
 Tween 295
 TweenInfo 295
 Tweening 277
 TweenService
 Create 295
U
 Übersetzung 431
 UDim2 268
 UIAspectRatioConstraint 281, 420
 UIGridLayout 278
 UIListLayout 279
 UIPageLayout 280
 UISizeConstraint 281
 UITableLayout 279
 UITextSizeConstraint 281
 Umgebungsgeräusch 306
 Union 414
 Update 461

UpdateAsync() 361
 UserOwnsGamePassAsync 444

V

Validierung
 serverseitige 374
 Variable 222
 erstellen 223
 globale 235
 Namensgebung 222
 Variablenstil 260
 Ventilator 98
 Verankerung 59, 416
 Verankerungspunkt 269
 Veröffentlichung 424
 Versatz 78
 Village 45
 Vorschaubild 457
 Voxel 417

W

Wait() 232
 Wasserfall 180
 Wasserfall-Effekt 181
 Wasserfläche 117
 Weld 87
 Whale 452
 while-Schleife 231
 Width 175
 Wiedergabeschleife 327
 Winkelgeschwindigkeit 100
 Wood, Joshua 19

X

Xbox-Richtlinien 424

Z

Zeitachse 319
 Zeng, Raymond 19
 ZIndex 270
 Zufallszahl 301, 405