Dokumentation: Videospiel programmieren (2D-Plattformer)

# Unity Benutzeroberfläche

Die Entwicklerumgebung von Unity kann beim erstmaligen Starten des Editors als komplex erscheinen. Um dem entgegenzuwirken, wird in diesem Abschnitt die Benutzeroberfläche des Editors vorgestellt, dazu zählen Menüs und Fenster. In Abbildung 4.8 ist die Benutzeroberfläche des Unity Editors mit den jeweiligen Fenstern, Schaltflächen und Inhalten zu sehen. Zudem bietet Unity die Möglichkeit die entwickelten Spiele für unterschiedliche Plattformen zu „bauen“. Diese Funktion ist hilfreich beim Testen des Videospiels auf verschiedenen Systemen.

Ein Bild, das Text, Screenshot, Elektronik enthält.

Automatisch generierte Beschreibung

Abbildung 4.8: Unity Editor Benutzeroberfläche

Der Einfachheit halber werden bei der Beschreibung des Editors die englischen Bezeichnungen genutzt.

1. Menüleiste: Bietet die grundlegenden Einstellmöglichkeiten des Editor und zusätzliche Funktionen.
2. Werkzeugleiste: Enthält Tools für die Bearbeitung von GameObjects in der Scene, sowie die Tasten, um die Scene zu testen.
3. Hierarchy: Die in der Scene vorkommenden GameObjects sind darin aufgelistet.
4. Scene: In diesem Fenster werden GameObejcts in der aktuellen Scene platziert, bearbeitet und die Umgebung gestaltet.
5. Game: In diesem Fenster wird das Spiel aus der Perspektive der Kamera (ebenfalls ein GameObject) dargestellt und gespielt.
6. Project/Console: In diesem Fenster werden alle im Projekt enthaltene Assets verwaltet. Über die Konsole kommuniziert Unity in Form von Hinweisen und Fehlermeldungen mit den Nutzern und Nutzerinnen.
7. Inspector: In diesem Fenster können die Eigenschaften (Components) des ausgewählten GameObject bearbeitet werden, z.B. Transform, Collider und hinzugefügte Scripts.

## GameObjects und Components

In Unity ist jedes im Spiel vorkommende Objekt ein GameObject. Für sich allein genommen macht ein GameObject nichts, es benötigt bestimmte Eigenschaften, um ein Charakter oder ein Effekt zu werden. GameObjects sind Behälter, die bestimmte Eigenschaften enthalten, wie Position, Materialien und Scripts. Diese Eigenschaften können als Komponenten (Components) hinzugefügt werden (siehe Abbildung 4.9) [(vgl. Unity Technologies 2016b)](#_CTVL0015cab624fc00a468fba609e6b3e6ea78e).

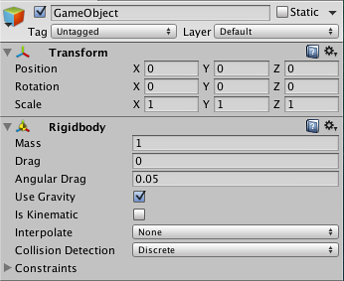


Abbildung 4.9: GameObject mit Transform und Rigidbody Komponente [(Unity Technologies 2016c)](#_CTVL00110b36de073d54a12a31dc4754491252f)

## Physics Engine

Eine Besonderheit von Unity ist die von Haus aus implementierte „physics engine“, welche die Entwickler und Entwicklerinnen beim Simulieren der Physik unterstützt. Darunter ist zu verstehen, dass Objekte korrekt beschleunigen und auf Kollisionen, Gravitation und anderer Kräfte reagieren. Hierfür stellt Unity unterschiedliche „physics engines“ zur Verfügung. Für diese Arbeit wird eine objektorientierte Engine genutzt [(vgl. Unity Technologies 2021k)](#_CTVL001cb4f35adc11a453cb89e8b493ab3a7be).

Die für diese Arbeit relevanten Components, die von der „physics engine“ beeinflusst werden sind:

* Rigidbody, gibt dem GameObject physikalische Eigenschaften und kann Kräfte und Momente aufnehmen [(vgl. Unity Technologies 2021l)](#_CTVL001ad86c489dd654c549b5c36fd8dfab928).
* Articulation Body erlaubt es physikalische Gelenkverbindungen zu erzeugen, wie Roboterarme und kinematische Ketten [(vgl. Unity Technologies 2021h)](#_CTVL00105556a3fa0ee40d282577e27e2f657f3).
* Joints verbinden Rigidbodies mit Gelenken untereinander [(vgl. Unity Technologies 2021j)](#_CTVL001ea1571c20eac49ab9ce910a72e606e86).
* Collider definieren die Form von Objekten und erlauben physikalische Kollisionen [(vgl. Unity Technologies 2021i)](#_CTVL001eb319786ad354d05b90b571adc7ea778)

Der Einsatz der „physics engine“ wird in Kapitel 6.3 erläutert.

## Scripts

Das Verhalten von „*GameObjects*“ wird von „*Components*“, die dem GameObject hinzugefügt werden kontrolliert. Die von Unity zur Verfügung gestellten Components sind vielseitig, aber bei bestimmten Aufgaben stoßen sie an ihre Grenzen. Ab diesem Punkt kommen Scripts zum Einsatz. Sie ermöglichen es GameObjects individuell anzupassen [(vgl. Unity Technologies 2016a)](#_CTVL00155ba4ed0206c45169b744f05cfa17e8e). In Abbildung 4.10 ist ein Standard Unity C#-Script dargestellt.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 4.10: Unity Script [(Unity Technologies 2016a)](#_CTVL00155ba4ed0206c45169b744f05cfa17e8e)

Die Besonderheit von Unity C# Scripts ist, dass sie sich von der *MonoBehaviour*-Klasse ableiten. MonoBehaviour enthält spezielle Methoden für die Programmierung von Spielen in Unity. Als Beispiel die *Update()*-Methode wird jeden Frame ausgeführt. In der Update()-Methode werden z.B. die Eingaben der Spielenden abgefragt [(vgl. Unity Technologies 2021d)](#_CTVL001866ed185e4e74e739ba56ffa7a85048f).

## ScriptableObjects

Ein *ScriptableObject* ist ein Datenspeicher zum Speichern von großen Mengen an Daten, unabhängig von der Klassen Instanz. Einer der wesentlichen Gründe für den Gebrauch von ScriptableObjects ist es den Verbrauch von Arbeitsspeicher zu reduzieren, indem Kopien von Werten verhindert werden. Die hauptsächlichen Gründe für den Einsatz von ScriptableObjects sind das Sichern von Daten während der Arbeit im Editor und das Speichern von Daten als Asset im Projekt während dessen Laufzeit [(vgl. Unity Technologies 2021e)](#_CTVL001cff6e7bee858470ab22b884de24b23c8).