Dokumentation: Videospiel programmieren (2D-Plattformer)

Inhaltsverzeichnis

[1 Projekt einrichten 1](#_Toc100673899)

[1.1 Unity Hub herunterladen und installieren 1](#_Toc100673900)

[1.2 Bei Unity registrieren 2](#_Toc100673901)

[1.3 Unity Editor und Komponenten installieren 2](#_Toc100673902)

[1.4 Projekt herunterladen 3](#_Toc100673903)

[1.5 Projekt im Editor öffnen 4](#_Toc100673904)

[2 Unity Benutzeroberfläche 5](#_Toc100673905)

[3 Szenen 6](#_Toc100673906)

[4 GameObjects und Components 6](#_Toc100673907)

[4.1 Prefabs (Standardteile) 6](#_Toc100673908)

[5 Programmierung 7](#_Toc100673909)

[5.1 C# Scripts 7](#_Toc100673910)

[5.2 Visual Scripting 7](#_Toc100673911)

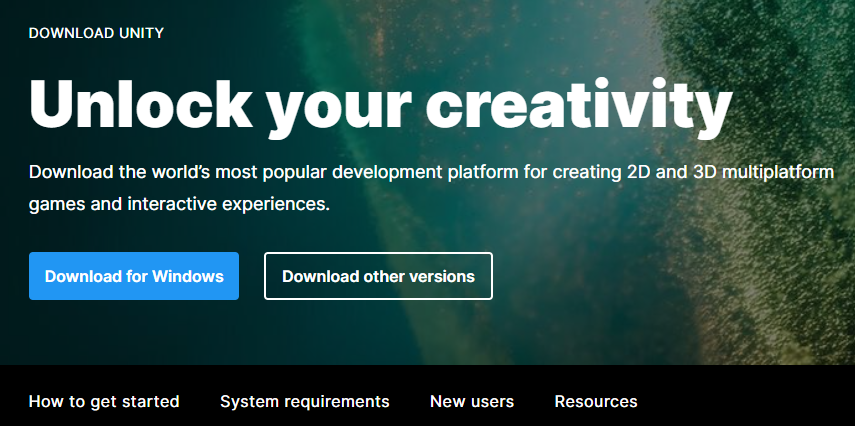
[5.3 ScriptableObjects 8](#_Toc100673912)

# Projekt einrichten

In den folgenden Abschnitten wird die Installation von Unity und dem Einrichten des Projekts beschrieben.

## Unity Hub herunterladen und installieren

Den Unity Hub können sie unter diesem Link <https://unity.com/download> herunterladen.



## Bei Unity registrieren

<https://id.unity.com/>

## Unity Editor und Komponenten installieren

Der Unity Editor wird wie folgt beschrieben installiert.

Im Fenster des Unity Hub unter dem Reiter „Installs“ finden Sie die installierten Versionen des Editors (siehe Abbildung 1.1). Bei der ersten Installation klicken Sie auf „Install Editor“, daraufhin öffnet sich ein weiteres Fenster (siehe Abbildung 1.2).

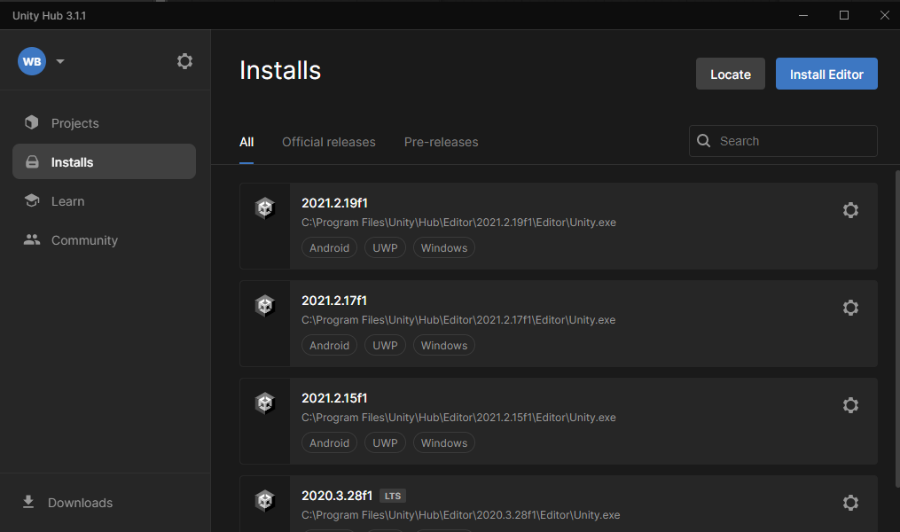


Abbildung 1.1: Unity Hub Installs

In diesem Fenster stehen ihnen verschiedene Versionen des Unity Editors zum Installieren zur Verfügung. In der Regel ist es sinnvoll die „Recommended version“ zu installieren, da dieses Project Visual Scripting nutzt sollte die Version 2021.1. oder aktueller installiert werden.

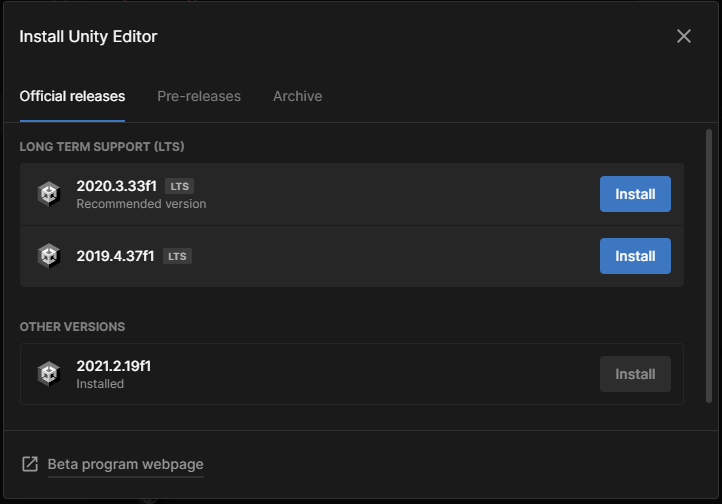


Abbildung 1.2: Install Unity Editor

Klicken Sie auf „Install“ und daraufhin öffnet sich erneut ein weiteres Fenster (siehe Abbildung 1.3). In diesem Fenster können zusätzliche Module für den Unity Editor installiert werden, diese ermöglichen im Nachhinein den Export des Spiels für eine bestimmte Plattform. Folgende Module sind für das Projekt zu empfehlen.

Im Dropdown Menü „DEV TOOLS“ ein Häkchen bei „Microsoft Visual Studio Community 2019“ setzen. Alternativ zu Visual Studio kann Visual Studio Code ( <https://code.visualstudio.com/> ) oder JetBrains Rider IDE ( <https://www.jetbrains.com/de-de/rider/> )genutzt werden. Im Dropdown Menü „PLATTFORMS“ finden sie die Module für die jeweiligen Plattformen, für die das Spiel exportiert werden soll. Auf einem Windows Gerät ist es sinnvoll die Module „Universal Windows Platform Build Support“ und „Windows Build Support (IL2CPP)“ auszuwählen.

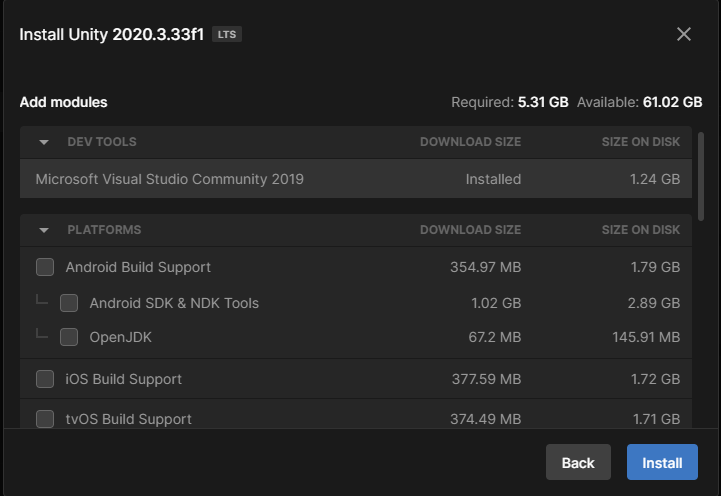


Abbildung 1.3: Unity Editor add modules

Es ist es gewünscht das Spiel z.B. für ein Android Smartphone zu exportieren kann das Modul „Android Build Support“ sowie die Untermodule „Android SDK&NDK Tools“ und OpenJDK“ installiert werden. Für eine Web-Anwendung wird das Modul „WebGL Build Support“ benötigt.

Klicken sie nun auf „Install“ um den Editor mit den Modulen herunterzuladen und zu installieren.

## Projekt herunterladen

Den Projektordner finden sie unter diesem Link <https://github.com/WadimBrakowski/Schuelerlabor-2D> . Klicken sie auf „Code“ und dann haben sie die Möglichkeit das Projekt zu klonen oder wenn sie nicht mit Git vertraut sind, können sie es unter „Download ZIP“ herunterladen und dann auf ihrem System entpacken (siehe Abbildung 1.4).

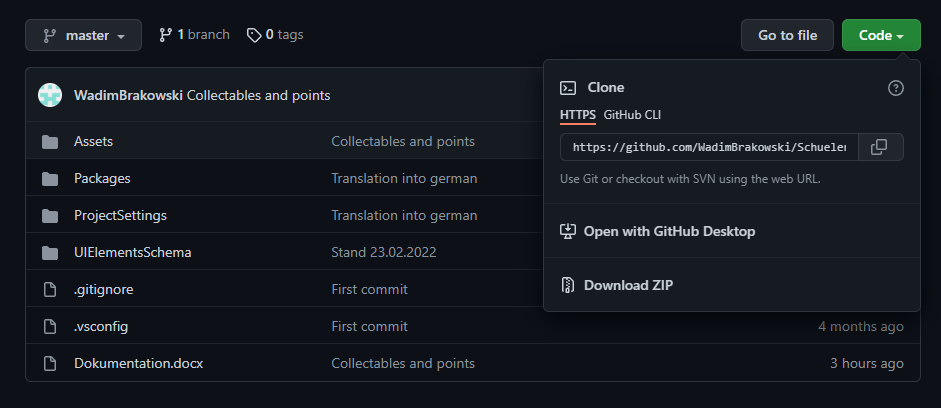


Abbildung 1.4: GitHub Projekt

## Projekt im Editor öffnen

Unter dem Reiter „Projects“ werden die Projekte aufgelistet. Um ein neues Projekt zu beginnen, klicken sie auf „New project“. In diesem Fall soll ein bestehendes Projekt geöffnet werden daher klicken sie auf „Open“ und dann auf „Add project from disk“ und wählen sie den Projektordner aus (siehe Abbildung 1.5). Daraufhin wird das Projekt gestartet, dies kann einen Moment dauern.

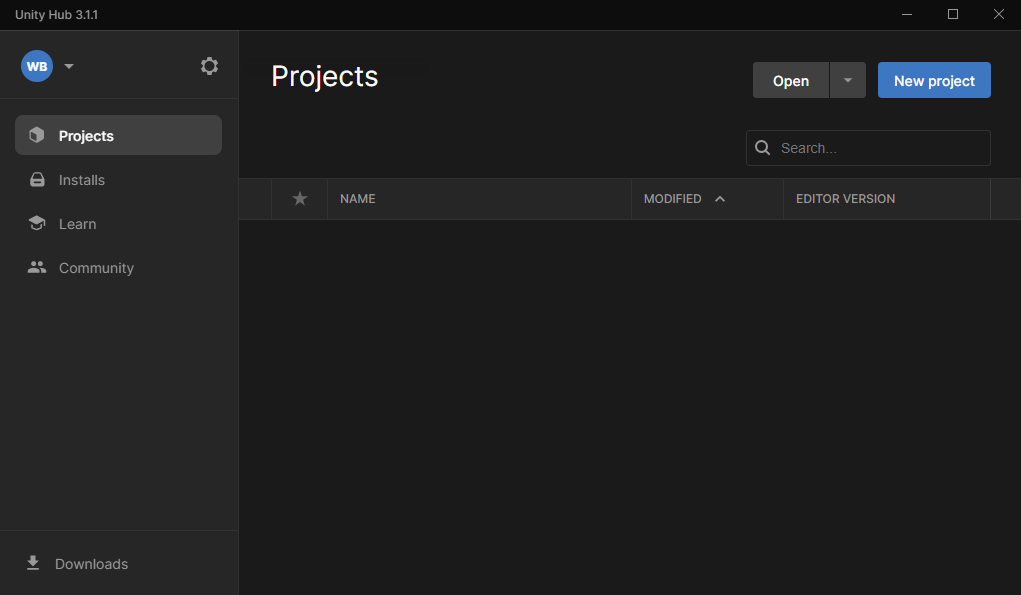


Abbildung 1.5: Unity Hub Projects Tab

Hinweis: Sollten nach dem Start „Errors“ angezeigt werden kann es helfen die „Packages“ zu aktualisieren. Dazu klicken Sie auf „Windwo“ und in der Liste weiter unten auf „Package Manager“ (siehe Abbildung 1.6).

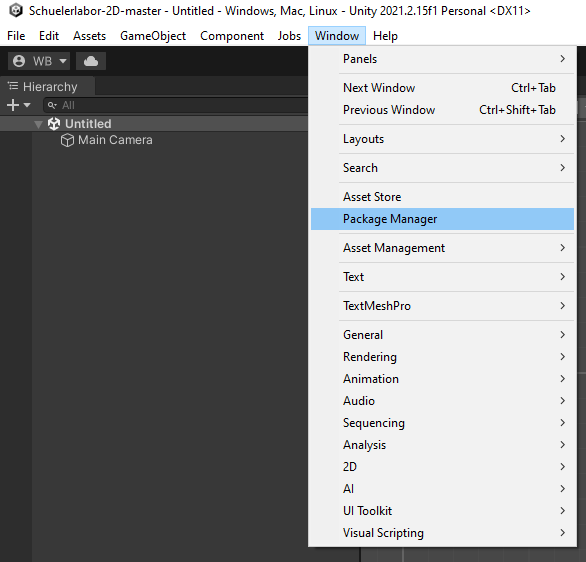


Abbildung 1.6: Unity Editor „Window“ Reiter

Es öffnet sich ein Fenster mit dem „Package Manager“. Dabei ist drauf zu achten, dass das Dropdown Menu „Packages“ auf „In Project“ steht, nun werden die im Projekt installierten Packages angezeigt. Ist eines der Packages mit einem Pfeil nach oben markiert ist für dieses Package ein Update verfügbar, welches über „Update to“ aktualisiert werden (siehe Abbildung 1.7).

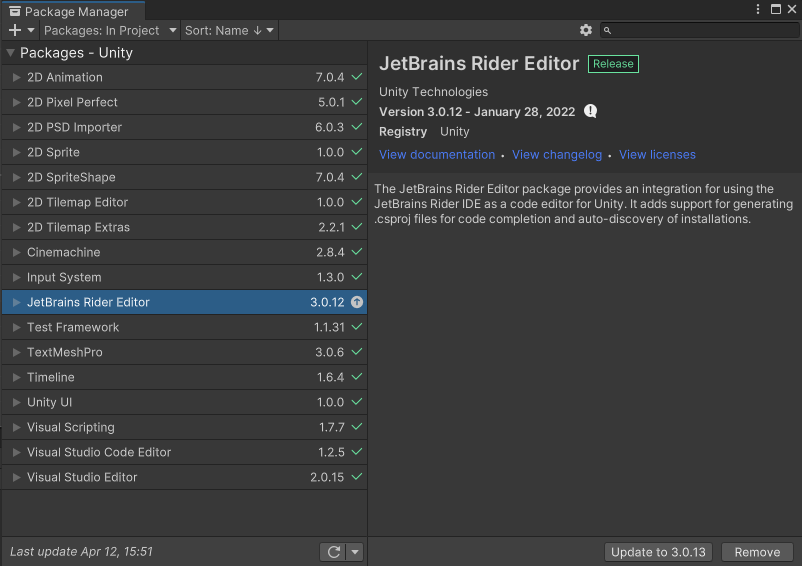


Abbildung 1.7: Unity Editor "Package Manager"

# 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 2.1 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 2.1: Unity Editor Benutzeroberfläche

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

1. Menüleiste: Bietet die grundlegenden Einstellmöglichkeiten des Editors 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.

# Szenen

Die Szenen in Unity kann man sich als Level oder als ein Spielabschnitt vorstellen. Sie dienen als Behälter für die Inhalte und Objekte für einen Spielabschnitt. Wie zuvor erwähnt werden die sich in der Szene befindenden Objekte in der Hierachy aufgelistet.

# 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 4.1) [(vgl. Unity Technologies 2016b)](#_CTVL0015cab624fc00a468fba609e6b3e6ea78e).

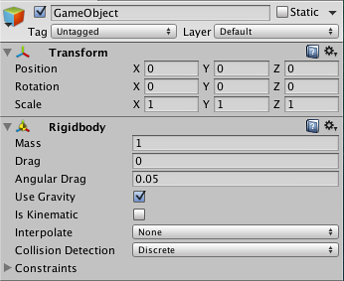


Abbildung 4.1: GameObject mit Transform und Rigidbody Komponente [(Unity Technologies 2016c)](#_CTVL00110b36de073d54a12a31dc4754491252f)

## Prefabs (Standardteile)

Soll ein GameObject öfter als einmal oder in verschiedenen Szenen verwendet werden, bietet es sich an von diesem GameObject ein Prefab anzulegen. Analog zum Maschinenbau sind Prefab eine Art Normteil. Diese werden im Vorfeld mit den gewünschten Eigenschaften versehen und können daraufhin in der Szene platziert werden. Prefabs werden wie folgt beschrieben erstellt:

1. GameObject in der Szene erstellen
2. Eigenschaften definieren
3. Das GameObject aus der Hierachy in einen in den Assers Ordner im Project Tab ziehen und Original Prefab anklicken.

# Programmierung

Die Programmierung in Unity für dieser Projekt ist objekt-orientiert, d.h. die Eigenschaften und das Verhalten, das ein GameObject haben soll, wird diesem zugewiesen. Dies kann mithilfe von C# Scripts und/oder durch eine visuelle Programmierung realisiert werden.

## C# 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 5.1 ist ein Standard Unity C#-Script dargestellt.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 5.1: 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).

Um ein C# Script im Unity Editor zu erstellen klicken sie mit der rechten Maustaste unter dem Reiter „Project“ in das Feld und navigieren die Maus über „Create“ und klicken auf C# Script (siehe.

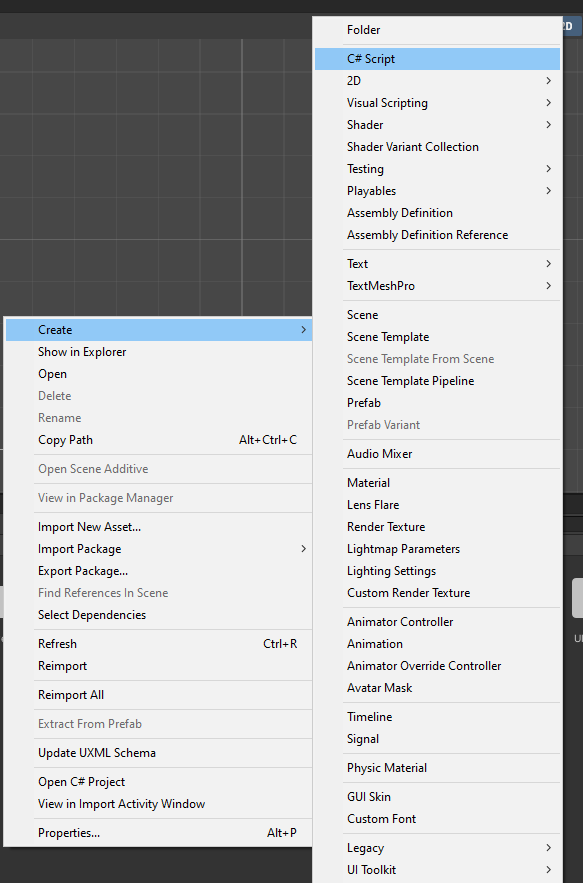
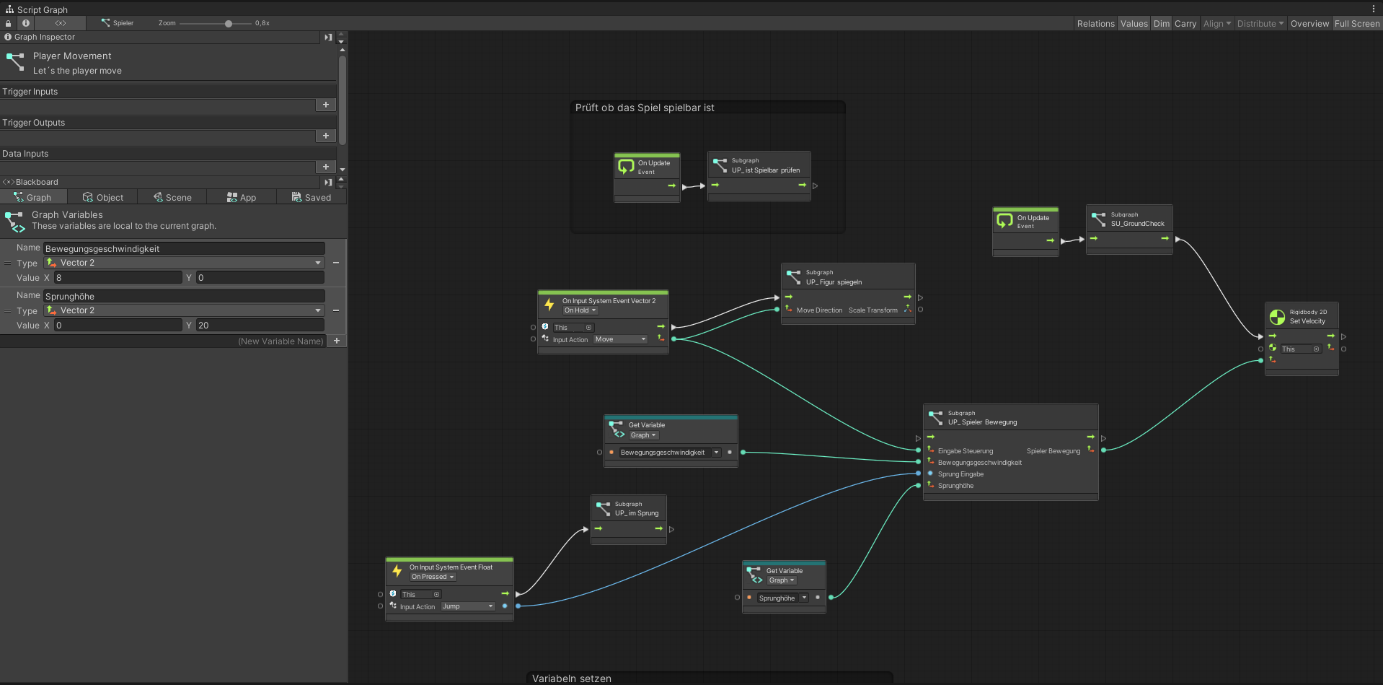


Abbildung 5.2: C# Script erstellen

## Visual Scripting



## 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).