

EVC „Übung 0“:

Vorbereitung:

Installieren Sie Unity auf dem PC, den sie in der Vorlesung benutzen werden. Empfehlenswert wäre, Unity über Unity Hub zu installieren zur schnelleren Verwaltung aller Projekte und der Unity Versionen:

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

Empfohlen: 2019.4.21f1 (LTS)

Wir werden auch ein 3D Modellierungsprogramm benötigen.

Zu empfehlen ist hier **Blender** (wird auch in der Vorlesung verwendet).

Wenn Sie bereits Expertise in einer anderen Software haben, können Sie diesen Schritt überspringen und ihre Software verwenden (Kein Support seitens der Vorlesung).

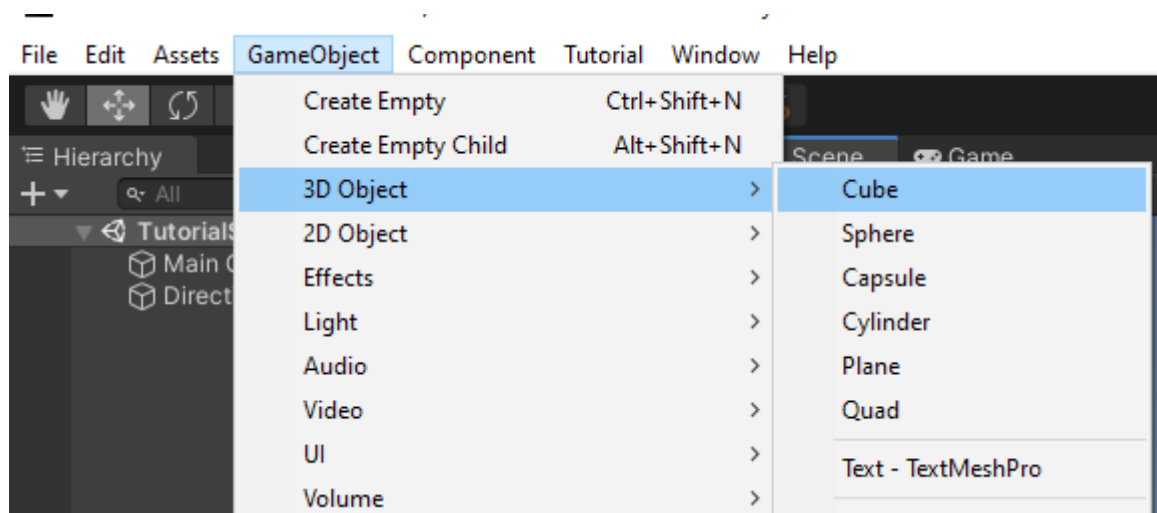
Link zu Blender: <https://www.blender.org/>

Übung (Basics in Unity):

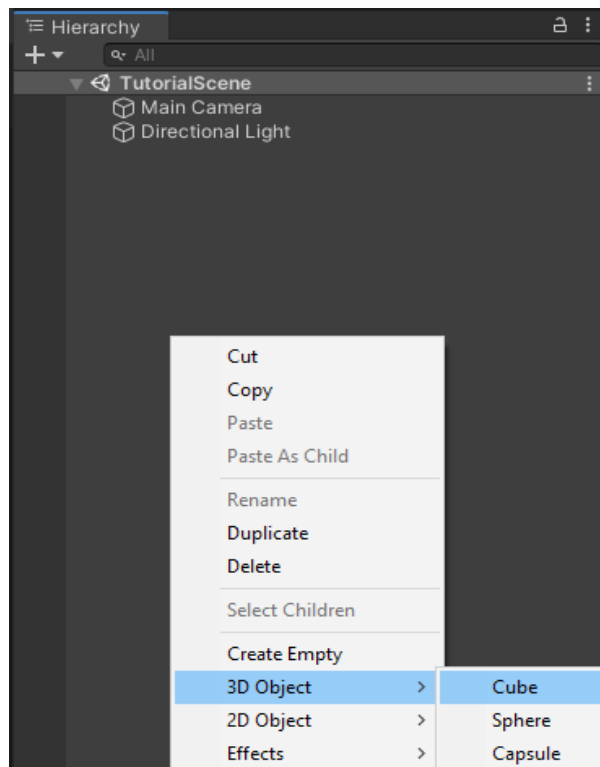
Erstellen Sie ein Projekt in Unity. (3D, kein Universal / High Definition Rendering Pipeline) und Arbeiten sie das Tutorial durch:

Unity Aufbau:

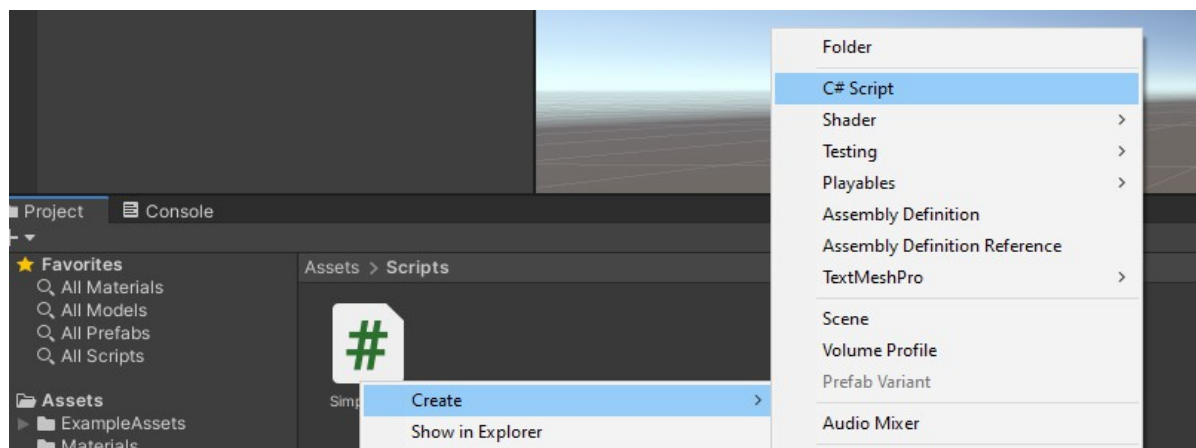
Erstellen Sie einen Würfel in der Standardszene:



Alternativ auch über Rechtsklick in der „Hierarchy“:

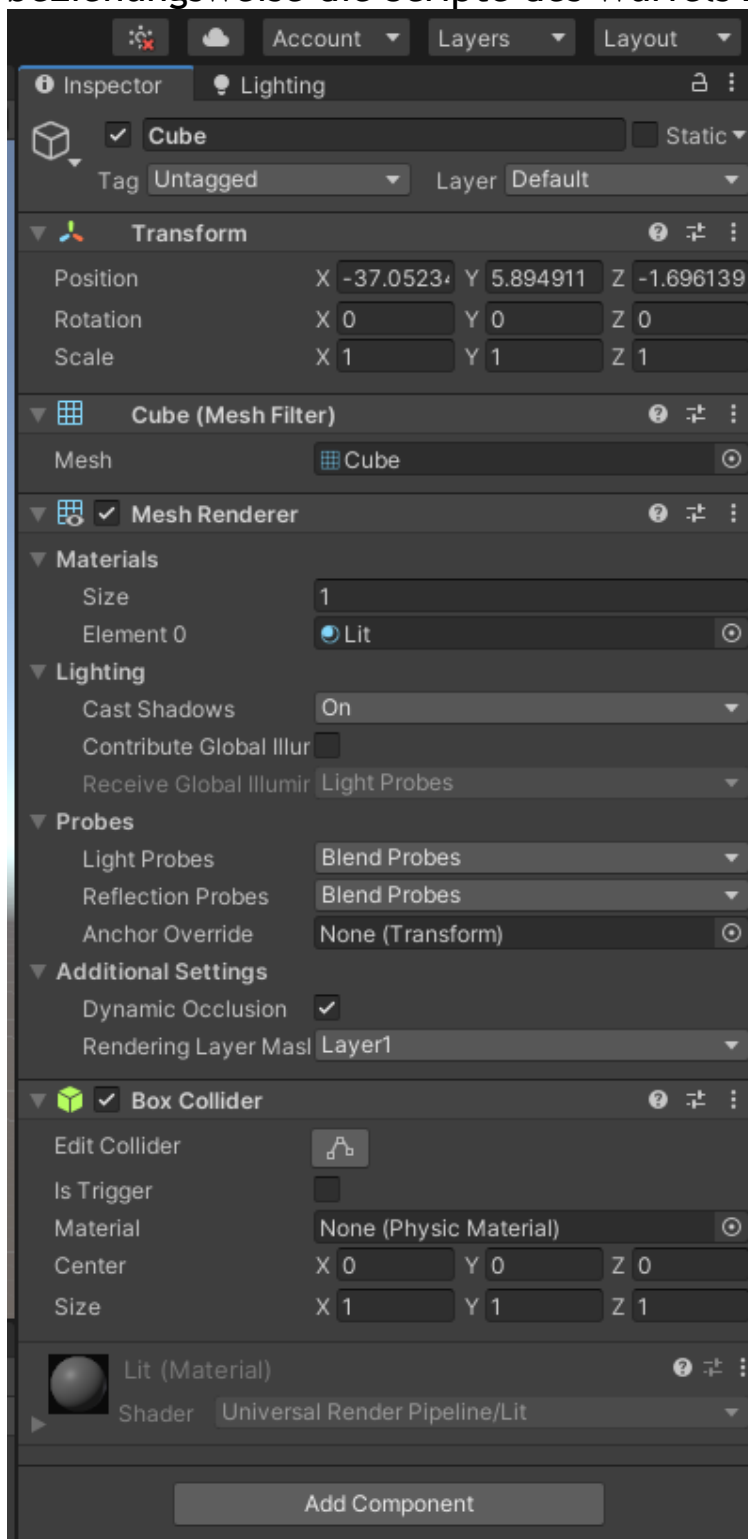


Erstellen Sie einen Ordner Scripts in dem Ordner Assets
(Standardanordnung: Unterer Bildschirmbereich).
In diesem Ordner erstellen Sie ein Script:



Nachdem wir im Folgenden den Würfel bewegen wollen, nennen wir das Script z.B. PlayerControlledMovement
(Sinn hinter der Namensgebung in Unity: In Unity werden Scripts als Komponenten / Verhalten betrachtet, z.B. würde eine Bombe das Script „Explosive“ erhalten um diesem die Fähigkeit / Verhalten eines Sprengstoffs zu geben)

Um dieses Script nun dem Würfel zuzuweisen, wählt man in der Hierarchy den Würfel durch Linksklick aus und nun sollten auf der rechten Seite des Bildschirms im „Inspector“ die Eigenschaften beziehungsweise die Scripte des Würfels zu sehen sein:

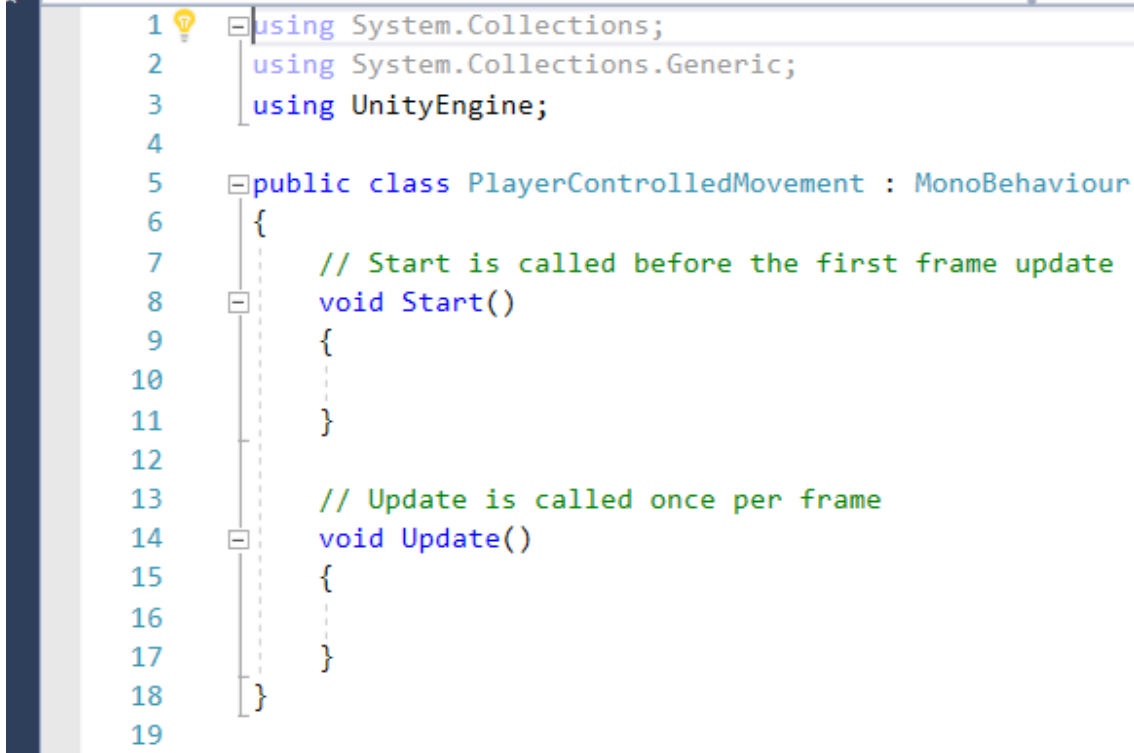


Das Script kann entweder durch Drag&Drop auf den Inspektor (wenn der Würfel gewählt ist) oder durch „Add Component“ hinzugefügt werden.

Das Script lässt sich entweder durch Doppelklick im Assets/Scripts Ordner öffnen, oder durch Doppelklick des Script Attributs der Script Komponente.

Script Aufbau (kurzer Exkurs):

Standardmäßig sieht ein Unity C# Script wie folgt aus:



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PlayerControlledMovement : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
19
```

Die Start Methode wird meist für Initialisierungszwecke verwendet und wird sehr früh ausgeführt (nicht als Erstes, aber sehr früh), wenn das Spiel startet und im Normalfall auch nur für einen Durchlauf.
(Ähnlich Arduino: setup())

Die Update Methode wird pro Frame, also bei 60fps, 60 mal ausgeführt. In diesem Bereich werden z.B. Bewegungen und Interaktionen gesteuert.
(Ähnlich Arduino: loop(); VV: Reaktives System)

Zum nachlesen, was es sonst noch gibt und in welcher Reihenfolge alles stattfindet:

<https://docs.unity3d.com/Manual/ExecutionOrder.html>

Ansonsten können Methoden und Variablen wie gewohnt deklariert werden.

Variablen:

Public Variablen sind im Inspektor in Unity sicht- und anpassbar.
Private Variablen nur mit dem Zusatz [SerializeField].

Eine Besonderheit bilden manche Variablen:

gameObject - bezieht sich immer auf das GameObject* auf dem das Script liegt.

transform - bezieht sich immer auf das **Transform des GameObjects auf dem das Script liegt.

*Jedes Objekt in Unity ist in erster Linie ein GameObject. Ein GameObject ist im Prinzip ein Container für alle weiteren Scripts und Eigenschaften. Jedes GameObject hat ein **Transform. Ein Transform beschreibt die Position, Drehung und Skalierung eines Objekts. (Ein GameObject kann andere Formen von Transforms, z.B. Rect Transforms (für UI/2D) besitzen, aber es besitzt immer ein Transform in irgendeiner Form).

-----Exkurs Ende-----

Würfel bewegen lassen:

Bevor wir den Würfel bewegen lassen können, sollten wir die Kamera noch so einstellen dass der Würfel auch sichtbar ist.

Um Unser Leben ein wenig einfacher zu gestalten zentrieren wir den Würfel im Koordinatenursprung.

Dazu den Würfel wählen in der Hierarchy und im Inspektor in der Komponente „Transform“ die Positionsvariablen auf 0 setzen (Alle 3). Mit einem doppelklick auf den Würfel in der Hierarchy springt die Ansicht zu dem gewählten Objekt.

Mit den Pfeiltasten, der mittleren und der rechten Maustaste können wir die Sicht richtig einstellen, damit der Würfel sich in der Bildschirmmitte befindet und auch etwas von uns entfernt ist (ca. 5-10% vom Bildschirm ausfüllend).

Jetzt können wir die Kamera in der Hierarchy wählen und dann im Reiter „GameObject“ des Unity Fensters „Align with View“ auswählen.

Alternativ funktioniert auch der Shortcut [strg + shift + F].

Damit passt sich die Kamera an die Position und Drehung unserer derzeitigen Sicht an.

Im Script:

Über *transform.position* bekommen wir einen 3 Dimensionalen Vektor, der die Position darstellt. Wenn wir diesem einen neuen Vektor zuweisen, weisen wir auch eine neue Position zu und der Würfel wird im nächsten Frame an der neuen Stelle angezeigt.

Hier eine Beispielzeile in der sich der Würfel auf der X Achse bewegt:

```
transform.position = new Vector3(transform.position.x + 2,  
transform.position.y, transform.position.z);
```

Je nach Einstellungen und Hardware kann es sein, dass der Würfel sich zu schnell aus dem Bild bewegt.

Grund dafür ist, dass die FPS Zahl sehr hoch ist und dementsprechend der Würfel sich schneller bewegt als dieser soll.

Man kann in solchen Fällen die Distanzen relativ an die FPS anpassen. Dafür multipliziert man den Wert mit *Time.deltaTime*.

```
transform.position = new Vector3(transform.position.x + 2 *  
Time.deltaTime, transform.position.y, transform.position.z);
```

Dadurch bewegt sich der Würfel mit 2 Einheiten pro Sekunde, unabhängig von der Framerate.

An der Stelle würde ich Sie darum bitten ein Wenig mit den Werten zu spielen und verschiedenes auszuprobieren.

Nachdem zu einem Spiel auch User Inputs verarbeitet werden müssen, müssen wir diese noch einbeziehen:

Input.GetKey(KeyCode.A)

liefert einen Bool zurück, ob in diesem Frame die „A“ Taste gedrückt wurde.

Damit ist eine grundlegende Steuerung des Würfels möglich.
Bitte setzen sie das um.

Nachdem diese Lösung nicht mit Controllern funktioniert und auch sonst etwas unschön ist, werden normalerweise Achsen zur Steuerung verwendet.

Input.GetAxis("Horizontal")

liefert einen Wert zurück, zwischen -1 und 1.
In der Mittelstellung und Deadzone des Controllers, beziehungsweise wenn weder A noch D auf der Tastatur betätigt werden liegt der Wert bei 0.

Passen sie ihre Steuerung so an, dass die Steuerung nicht mehr über KeyCodes geht, sondern über Achsen.

(Hinweis: Multiplikativ einberechnen, wie mit deltaTime)

Die Konfiguration der Achsen findet man in:

Edit > Project Settings > Input Manager

Dort finden sich auch die Strings, mit denen die Achsen angesprochen werden können.

