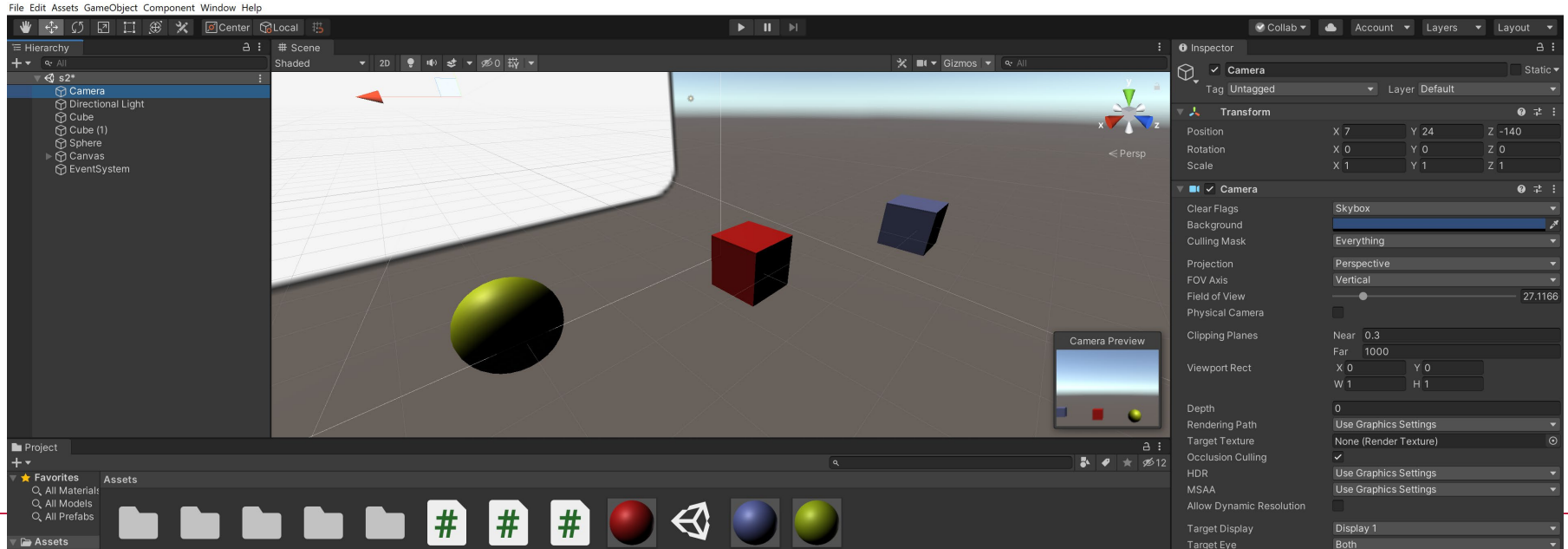
# Unity 2019.2.9f1

## Collision Detection

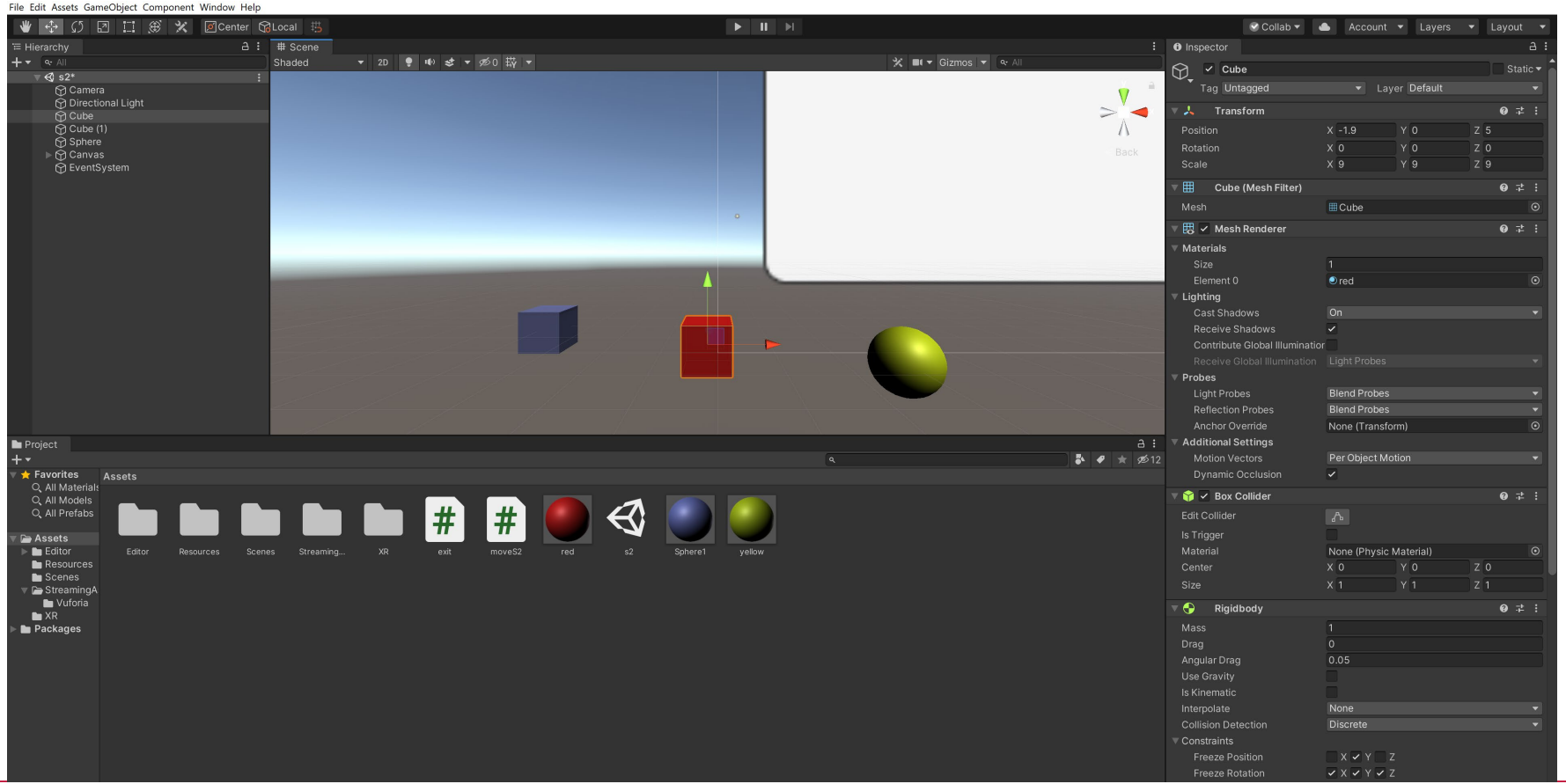
# Übung Collision Detection

**Spielidee:**

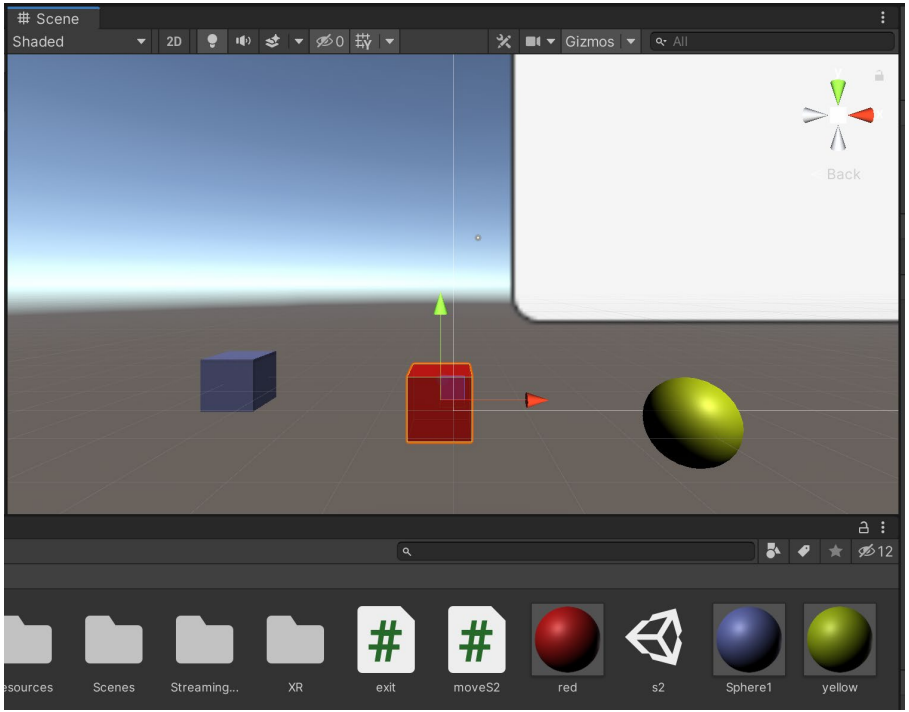
Es sind drei geometrische Objekte zu erstellen. Zwei Quader (cube) und eine Kugel (sphere). Jedes Objekt erhält eine eigne Farbe. Ein Objekt wird automatisch in Bewegung gesetzt und kann über die Pfeiltasten der Tastatur in X- und Z-Richtung (Horizontal) gesteuert werden. Wenn das steuerbare Objekt mit den beiden anderen Objekten kollidiert, dann soll dies registriert und gezählt werden. Die Anzahl der Kollisionen ist in einem Textfeld auszugeben.



# Übung zu Collision Detection



# Aufbau der Szene



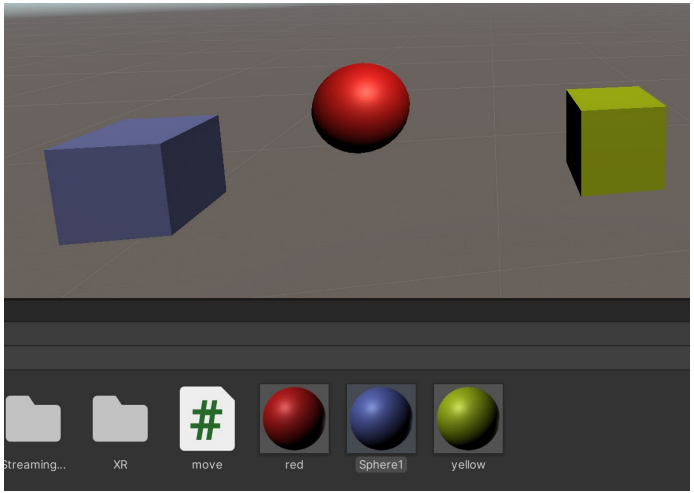

1. Der rote Würfel kann über die Pfeiltasten gesteuert werden. Hierzu verwenden wir das mitgelieferte C#-Programm.
2. Der blaue Würfel und die gelbe Kugel sind Objekte, die vom roten Würfel berührt werden können.
3. Die Berührungen sollen erkannt und gezählt werden, dies erledigt das C#-Programm.
4. Die Anzahl der Berührungen sollen in einem Textfeld ausgegeben werden (C#-Programm).

# Erstellen von zwei Cubes(Würfel) und einem Sphere(Kugel)

1. Objekte erstellen

Würfel: Menü → GameObject → 3D-Object → Cube

Kugel: Menü → GameObject → 3D-Object → Sphere




3. Objekte einfärben

Material erstellen: Menü → Assets → Create → Material

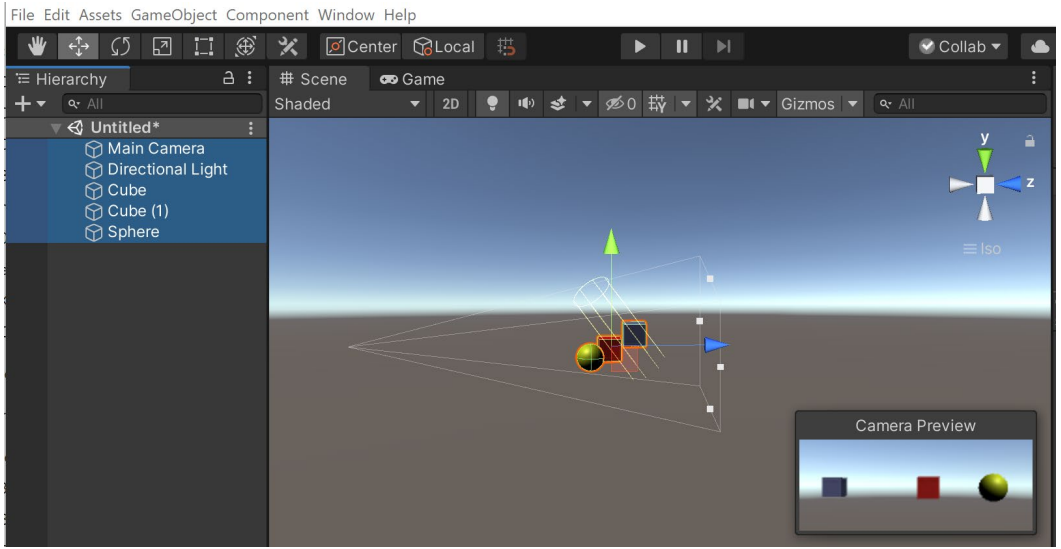
Das Material erscheint im Assets-Fenster. Wir vergeben einen Namen.

Die Farbeinstellung wird im Inspector-Fenster vorgenommen.

Ist die Farbe festgelegt, so kann das Material mit der Maus einfach auf das entsprechende Objekt gezogen werden.

# Objekte ausrichten

1. Testen, ob die Kamera die Objekte sieht. Dies geschieht, in dem die Main Camera in der Hierarchy angeklickt wird. Es erscheint ein kleines Fenster in der Scene mit der Kamerasicht. Sie können die Scene auch starten und kontrollieren, ob die Objekte für Ihre Kamera sichtbar sind.
2. Ausrichten der Kamera
   Mit der Maus oder dem Gismo wird die Welt so gedreht, dass die Kamera sichtbar ist. Eventuell benötigen Sie zwei unterschiedliche Ansichten zur Kontrolle, ob die Kamera die Objekte im Focus hat.



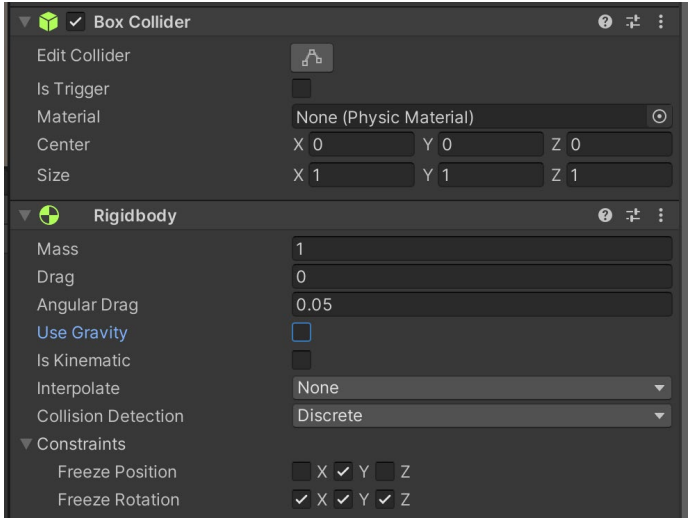
# Collision Detection

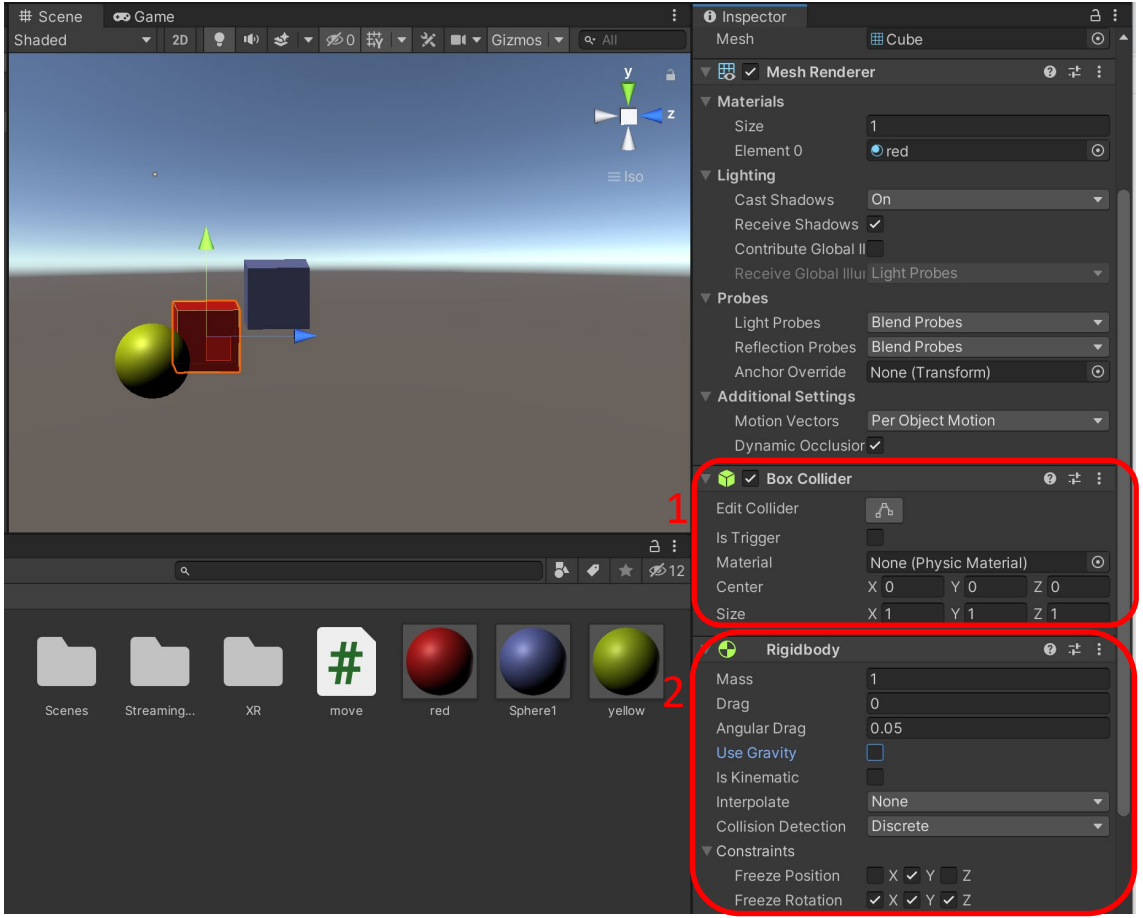
In der Unity-Physics-Engine integriert ist die „Collision Detection“.

Erst wenn man dem GameObject eine **Box-Collider**-Komponente hinzufügt, werden Berührungen mit anderen Collidern registriert. Die Physics- Engine ist wie eine unsichtbare Welt im Hintergrund, in der nur Objekte vorkommen, die mit entsprechenden Komponenten ausgerüstet sind.

GameObjects mit einem Collider, die nicht nur an einer Position von Anfang bis zum Ende verharren, sondern sich auch **in Bewegung setzen**, müssen zusätzlich eine **Rigidbody**-Komponente haben. Diese dient Unity bei der Optimierung der Performance von Collidern und ist Voraussetzung, um eine Kollision aus der Bewegung heraus auszulösen. Wenn zwei Objekt aufeinandertreffen, muss also mindestens eines der beiden einen Rigidbody haben, damit die Events im Code ausgelöst werden.

Objekt aktivieren. Über ***Component → Physics → Rigidbody*** einschalten.






# Rigidbody

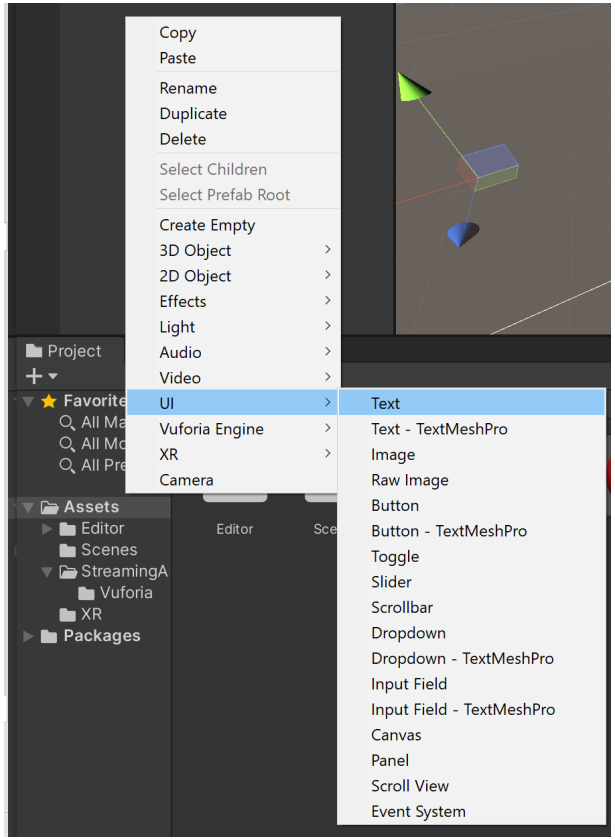
Einstellungen ***Rigidbody*** des bewegl. Cube:

1. ***Box Collider*** muss aktiv sein
2. ***Use Gravity*** deaktivieren und ***Constraints*** wie im Bild aktivieren

# Ein Textfeld erzeugen

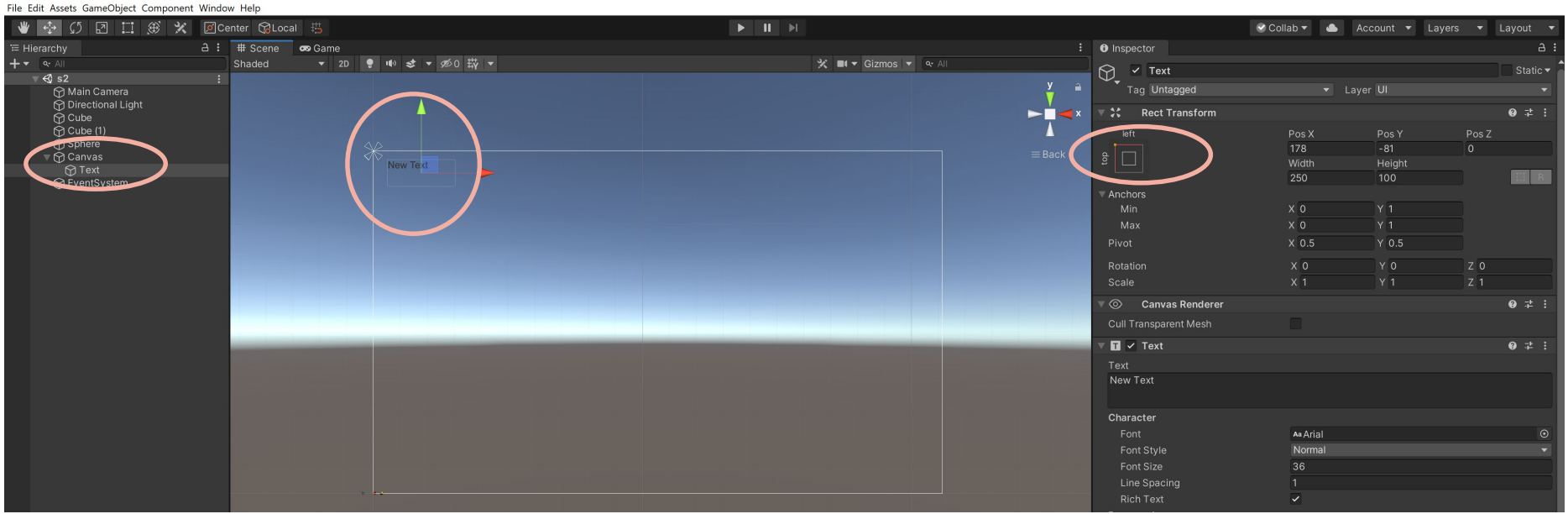
1. In der Hierarchy mit der rechten Maustaste
   das Menü-Fenster öffnen und
   anschließend **UI** und **Text** auswählen.

2. Die Position des Textfeldes durch den Sceneplayer testen.
   Das Textfeld wird im Kamerafenster leider nicht angezeigt.



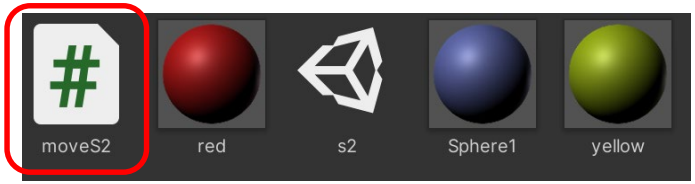
# Das Textfeld ausrichten

Das Textfeld befindet sich innerhalb einer größeren Canvas-Fläche. In dieses Textfeld soll die Anzahl der Kollisionen eingetragen werden.



# Die Programmierung

1. Über Menü → Assets → Create → C# Skript erzeugen sie ein neues C#-Programm. Geben Sie diesem Programm, das im Assets-Fenster aufgeführt wird, einen Namen




2. Durch einen Doppelklick auf das #-Symbol im Assets-Fenster öffnet sich ein Editor (Visual Studio) mit einem rudimentären Programm-Code, der nun erweitert werden muss.
3. Ändern Sie den Klassennamen in den Namen, den sie für dieses Programm im Asset-Fenster vergeben haben, ohne die Endung „.cs“!
4. Setzen sie nun die im Folgenden gegebenen Programmbausteine ein.
5. Das Programm muss dem zu bewegenden Objekt (Roter Würfel) zugewiesen werden. Dies geschieht, in dem es mit der Maus aus dem Assets-Fenster auf das Objekt in der Hierarchy gezogen und dann losgelassen wird. Danach ist das Programm am unteren Ende des Inspectors des Objekts zu sehen.

# C#-Programm

**Teil 1:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class move : MonoBehaviour
{
    public Text hits;
    public float colls = 0;
    public float baseVelocity = 5F;
    private Rigidbody myRigidbody = null;

    // Start is called before the first frame update
    void Start()
    {
        myRigidbody = GetComponent<Rigidbody>();
        myRigidbody.velocity = transform.forward * baseVelocity;
    }
```

# C#-Programm

**Teil 2:**

```
// Update is called once per frame
    void Update()
    {
        float rotation = 0F;
        if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            rotation = -90F;
        }
        else if(Input.GetKeyDown(KeyCode.RightArrow))
        {
            rotation = 90F;
        }
        if (rotation != 0F) {
            transform.Rotate(0F, rotation, 0F);
            myRigidbody.velocity = transform.forward * baseVelocity;
        }
    }
```

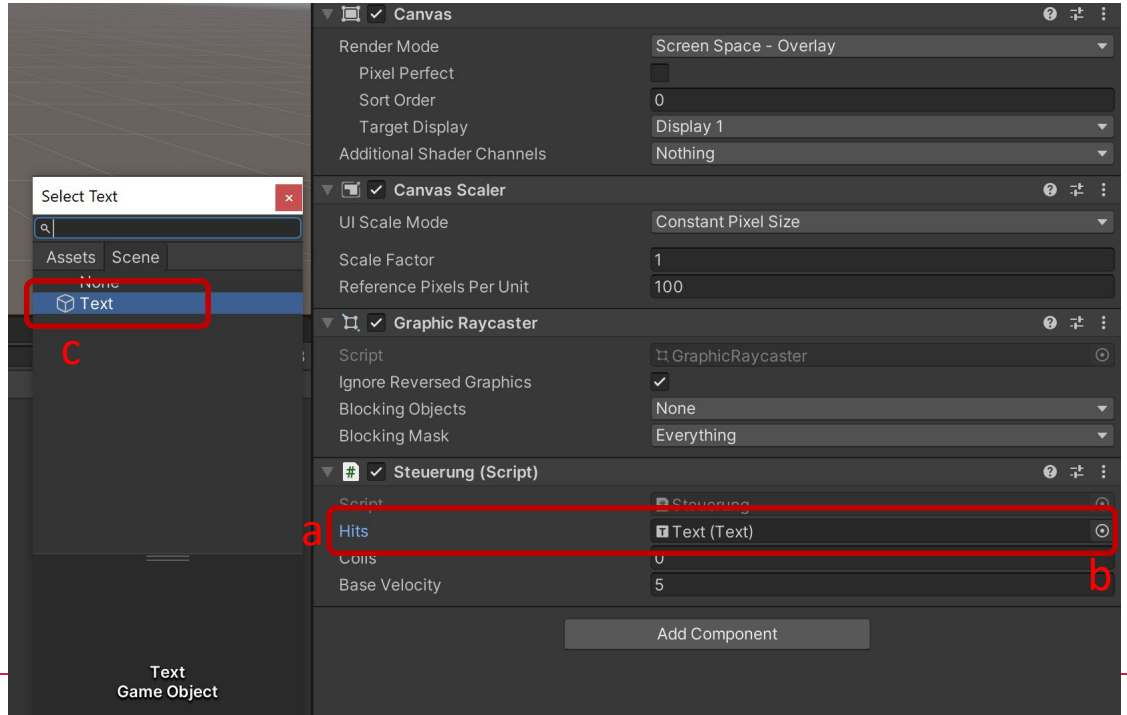
# C#-Programm

**Teil 3:**

```
void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.name == "Cube")
        {
            colls++;
            Debug.Log(colls);
          // Wenn das kollidierende Objekt den Namen "Cube" hat, wird dieser Code ausgeführt.
        }
        if (collision.gameObject.name == "Sphere")
        {
            colls++;
            Debug.Log(colls);
          // Wenn das kollidierende Objekt den Namen „Sphere" hat, wird dieser Code ausgeführt.
        }
        hits.text = "Contact = " + colls;
    }
}
```

# Programm mit dem beweglichen Objekt verbinden

1. **Nach dem Erstellen des C#-Skripts wird das Skript aus dem Assets-Bereich auf den zu steuernden Cube im Hierarchy- oder Scene-Fenster gezogen. Daraufhin erscheint das Skript im Inspector des Objekts Cube.**
2. Sobald das Skript und das Textfeld erstellt sind, wird der Variablen „Hits“ (a) das Textfeld zugewiesen. Wenn der kleine Kreis (b) rechts des Hits-Textfeldes mit dem Eintrag „None (Text)“ geklickt wird, können Sie das Skript dem Textfeld mit dem Namen Text im Fenster „Select Text“ (c) zuweisen.



# Build And Run

Das Programm Compilieren und Ausführen.

Menü → File → Build And Run

Im Console-Fenster können nun Fehlermeldungen und Ausgaben angesehen werden, die im Programm mit **`Debug.Log(Variable);`** oder **`Debug.Log(text);`** eingerichtet werden müssen.

Um eine Anwendung weiterzugeben, müssen die folgenden Dateien eines Projektes enthalten sein:

| | |
|---|---|
| Ordner: | collission_detection_a_Data |
| Ordner: | MonoBleedingEdge |
| Datei: | collission_detection_a.exe |
| Datei: | UnityPlayer.dll |

# Einbau eines Exit-Buttons

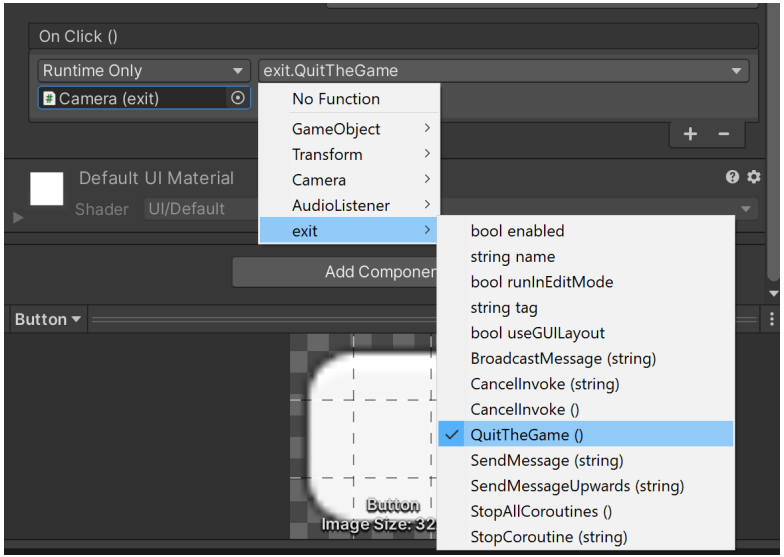
1. Im Fenster Hierarchy mit der rechten Maustaste auf den Canvas-Eintrag klicken. Danach auf UI und dann auf Button klicken.
2. Die Größe des Buttons einstellen, damit er sichtbar und der Button-Text lesbar ist.
3. Den Button auf der Bildschirmoberfläche ausrichten über die "Anchor Presets". Das C#-Programm zum Ausstieg aus dem Programm mit **Assets → Create → C# Script** erzeugen und den folgenden Programmcode einfügen.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class exit : MonoBehaviour
{
    public void QuitTheGame()
    {
        Application.Quit();
    }
}
```

# Das Exit-Programm richtig zuweisen!

1. Ziehen Sie das Exit-Programm aus dem Assets-Fenster auf die Kamera.
2. Aktivieren Sie den Button und klicken sie im Fenster Inspector auf das „+“ unter On Click ().
3. Klicken Sie auf den kleinen runden Kreis mit der Bezeichnung "None (Object)".
4. Wählen Sie in dem erscheinenden Fenster den Reiter „Scene“.
5. Wählen sie in der Liste „Camera“ aus
6. Klicken sie auf das Feld rechts von „Runtime Only“.
7. Fahren Sie mit der Maus auf das richtige Programm und wählen sie im aufgehenden Fenster die gewünschte Funktion aus.
8. Starten sie "Build And Run" und testen sie die Funktion.



# Literatur

Chittesh J. (2015) Das Unity-Buch 2D- und 3D-Spiele entwickeln. Heidelberg, Dpunkt-Verlag.

Seifert C. (2014) Spiele entwickeln mit  Unity-Buch. München, Hanser-Verlag.

# Kontakt:

**Prof. Martin Schober |** Informations- und Medientechnik

**Hochschule Karlsruhe - Technik und Wirtschaft**
Fakultät für Informationsmanagement und Medien

**Postanschrift: Postfach 24 40, 76012 Karlsruhe**
Besucheranschrift: Amalienstr. 81-87 | 76133 Karlsruhe | Raum AM 113
fon +49 (0)721 925 - 2990 | fax +49 (0)721 925 -1125
martin.schober@h-ka.de | www.technischeredaktion.com