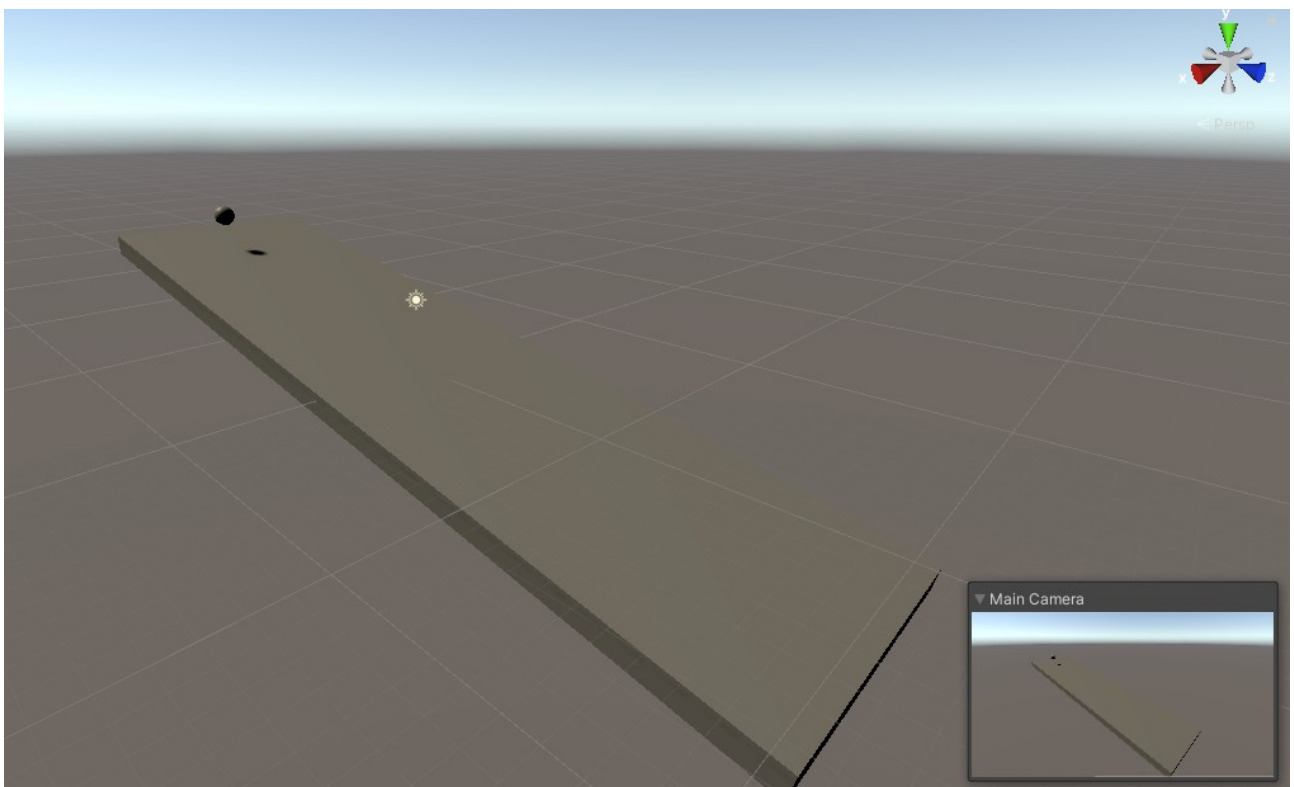


Übung 1: Physik, Collider und Trigger

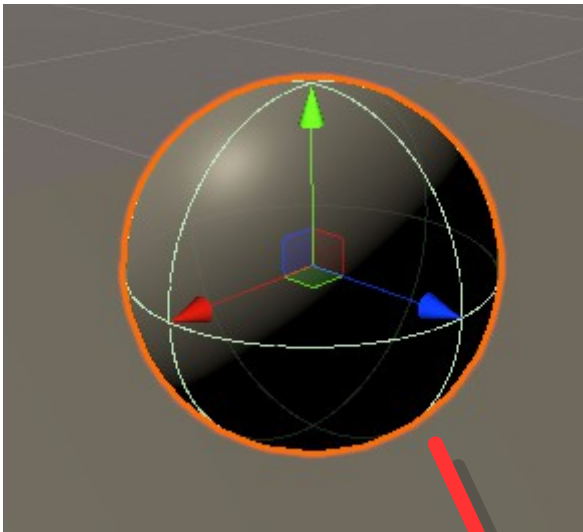
Vorbereitung:

- Erstellen Sie ein neues Projekt (oder eine neue Szene im Übungsprojekt) und erschaffen sie einen Würfel.
- Skalieren Sie den Würfel so, dass dieser zu einem breiten Brett wird.
- Kippen Sie das Brett leicht, sodass es eine Rampe darstellt (10° Neigung reicht).
- Erstellen Sie eine Kugel und platzieren sie diese am hohen Ende der Rampe.
- Platzieren Sie die Kamera so, dass diese Alles erfassen kann.

Das sollte in etwa so aussehen:

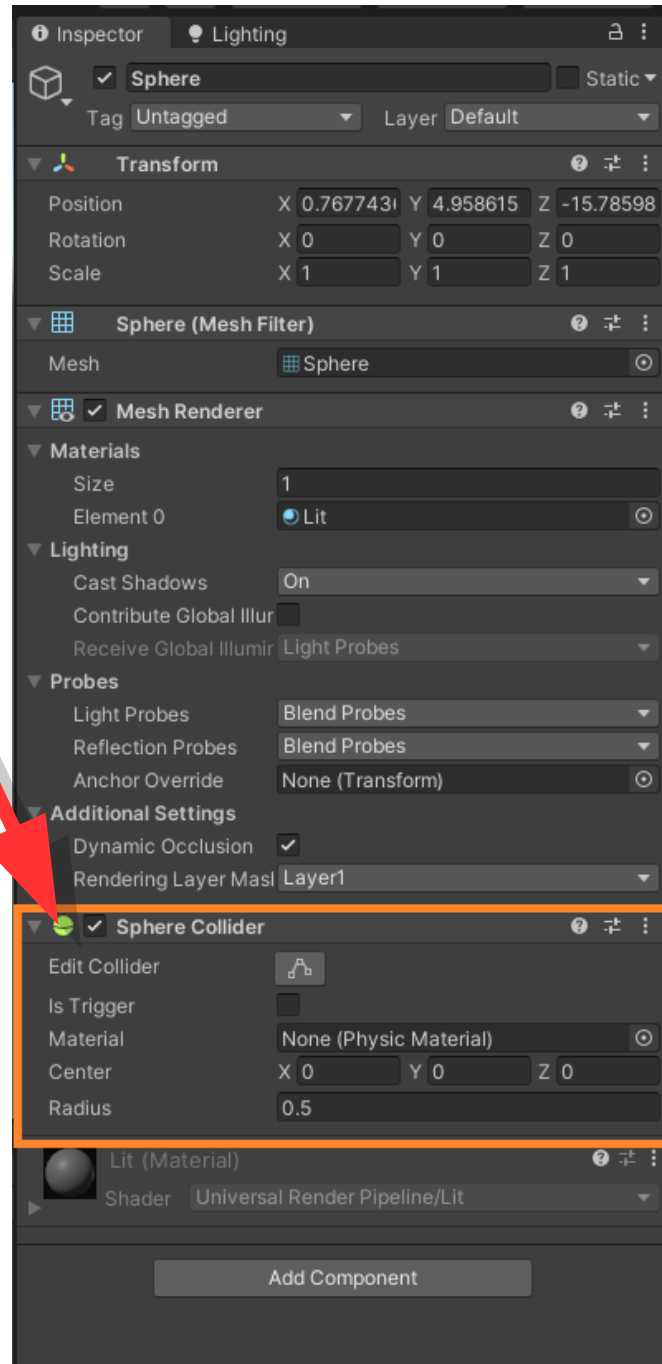


Exkurs über Collider:



Ein Collider liefert der Physics Engine Informationen darüber, wie ein Objekt sich physikalisch zu verhalten hat und stellt dessen „harten“ Körper dar. Ohne Collider reagiert das Modell wie ein Geist / Hologramm.

Collider sollten möglichst eine simple Struktur aufweisen. Daher werden die Proportionen oftmals nur approximiert. Mögliche Collider in Unity sind: (~aufsteigend komplex)
Cube Collider,
Sphere Collider,
Capsule Collider,
Terrain Collider,
Mesh Collider



Über ein optionales „Physics Material“ können dem Collider zusätzliche Eigenschaften zugewiesen werden, wie etwa die Oberflächenreibung und die Elastizität der Kollision.

Teil 1:

In diesem Teil soll eine Lackier Station für unseren Würfel entstehen, der unsere Kugel umlackiert.

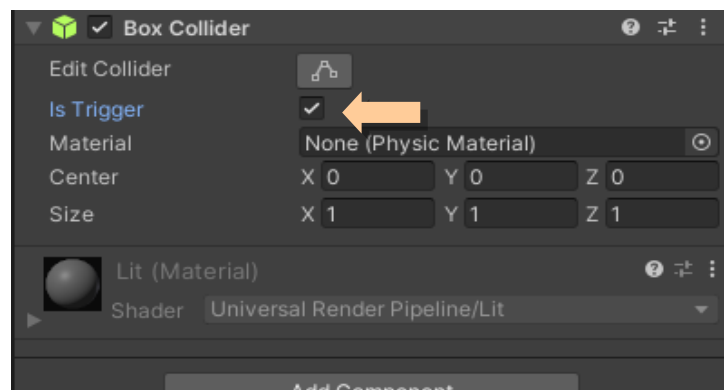
Fügen Sie der Kugel eine „Rigidbody“ Komponente hinzu.
→ Die Kugel sollte den Abhang herabrollen. (Auf „Play“)

(Die Rigidbody Komponente gibt dem Objekt die Möglichkeit von Physischen Kräften und Kollisionen betroffen zu sein)

Platzieren sie einen Würfel auf dem Rollweg der Kugel (mit Boxcollider).
Markieren sie diesen als Trigger.

→ Die Kugel Rollt durch diesen Trigger hindurch (Auf „Play“)

erstellen sie ein Script „Painter“ (oder ähnlich) und weisen Sie dieses dem Würfel zu.

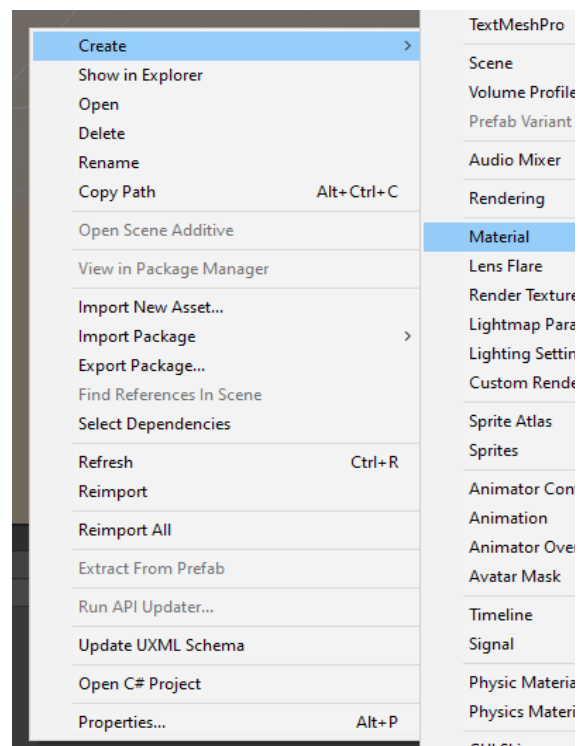


Erstellen Sie zwei neue „Materials“ über das Rechtsklick Menü (im Asset/Datei Explorer):

Und benennen Sie diese nach den Farben, die sie diesen vergeben werden z.B. Lila und Rot.

Weisen Sie diesen die Farbe über das Color-Picker Tool (Rechts im Inspector bei „Base Map“) zu.

Über Drag&Drop geben sie der Kugel eines der neuen „Materials“.



Im Code kann durch die public Methode „OnTriggerEnter“:

<https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html>

Die Kollision erkannt werden. Als Parameter bekommt die Methode die Instanz des anderen Kolliders. (Bei der Kollision von unserer Kugel mit dem Painter Würfel, wäre das die Kugel)

Auf Basis dieser Instanz kann auf andere Komponenten zugegriffen werden:

<https://docs.unity3d.com/ScriptReference/GameObject.GetComponent.html>

Die Komponente, die wir dabei brauchen nennt sich „Renderer“.

Diese besitzt ein Attribut namens Material. Mit überschreiben dieses Felds können wir die Farbe (und theoretisch einiges mehr ändern)

Sinnvollerweise braucht es dazu ein Material, das wir zuweisen möchten. Dazu fehlt noch eine Variable der wir im Inspector eine Variable zuweisen können. (Hinweis: public, oder private mit [SerializeField])

Schreiben Sie die Methode und weisen sie im Inspector das Material dem Script zu. (Beispiel auf nächster Seite)

```
C# Assembly-CSharp Painter
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Painter : MonoBehaviour
6 {
7     [SerializeField] private Material materialToPaint;
8
9     public void OnTriggerEnter(Collider other)
10    {
11        other.GetComponent<Renderer>().material = materialToPaint;
12    }
13 }
14
```

Teil 2:

(erstes Spiel der Vorlesung)

Die Kugel soll nach links und rechts steuerbar werden.

Es wird mehrere „Painter“ Würfel geben, die der Kugel ein neues Material zuweisen. (Sinnvollerweise haben diese Würfel auch dieses Material).

Am Ende des „Bretts“ soll es einen Checker geben, der die Farbe prüft und bei falscher Farbe die Kugel wieder nach oben Teleportiert.

Vorbereitung:

Drehen Sie das Brett so, dass es in Y Rotation gerade (mehrfaches von 180° (oder 0)) steht, falls es das noch nicht ist.

Wie Input entgegengenommen wird, wurde in der letzten Übung bereits behandelt. Ähnlich funktioniert es hier auch.

Dieses Mal werden wir aber Kräfte benutzen.

Kräfte können über AddForce auf den Rigidbody der Kugel übertragen werden. [AddForce](#)

Machen Sie die Kugel steuerbar.

(Kleiner Hinweis: hier arbeiten wir mit der Physics Engine, deswegen sollten wir statt in Update in FixedUpdate arbeiten)

Sinnvollerweise sollten Sie den Input mit 10 multiplizieren, damit dieser auch ausreichend wirksam ist.
(Unter Umständen müssen Sie diesen negativ gestalten um die Steuerung zu invertieren)

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BallController : MonoBehaviour
6  {
7      Rigidbody m_rigidbody;
8
9      // Start is called before the first frame update
10     void Start()
11     {
12         m_rigidbody = gameObject.GetComponent<Rigidbody>();
13     }
14
15     // Update is called once per frame
16     void FixedUpdate()
17     {
18         m_rigidbody.AddForce(Vector3.right * Input.GetAxis("Horizontal") * 10);
19     }
20 }
21
```

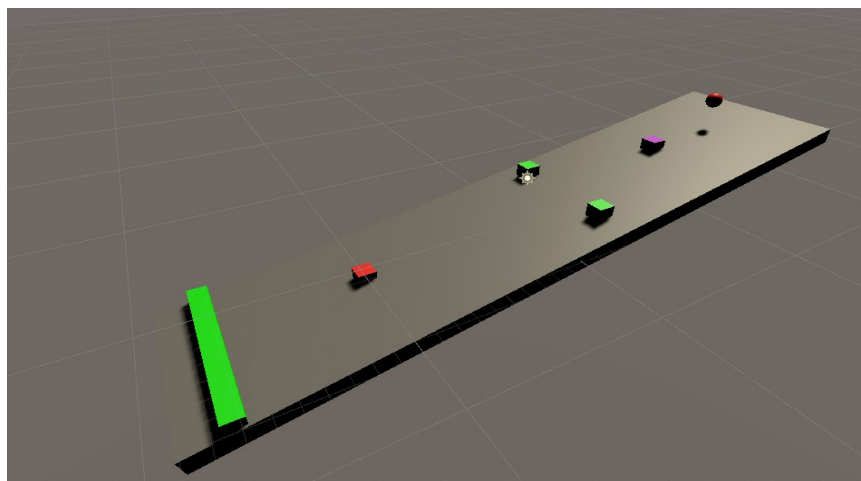
Jetzt sollte sich der Würfel nach links und rechts bewegen lassen.

Erstellen Sie noch mehr Materials und kopieren Sie die Painter Würfel. Verteilen sie diese auf der Rampe, um einen Parcour zu bilden. Weisen Sie den Würfeln die neuen Materials zu (Script und auf dem Modell).

Erstellen sie noch einen Würfel und skalieren sie diesen auf die breite des Bretts.

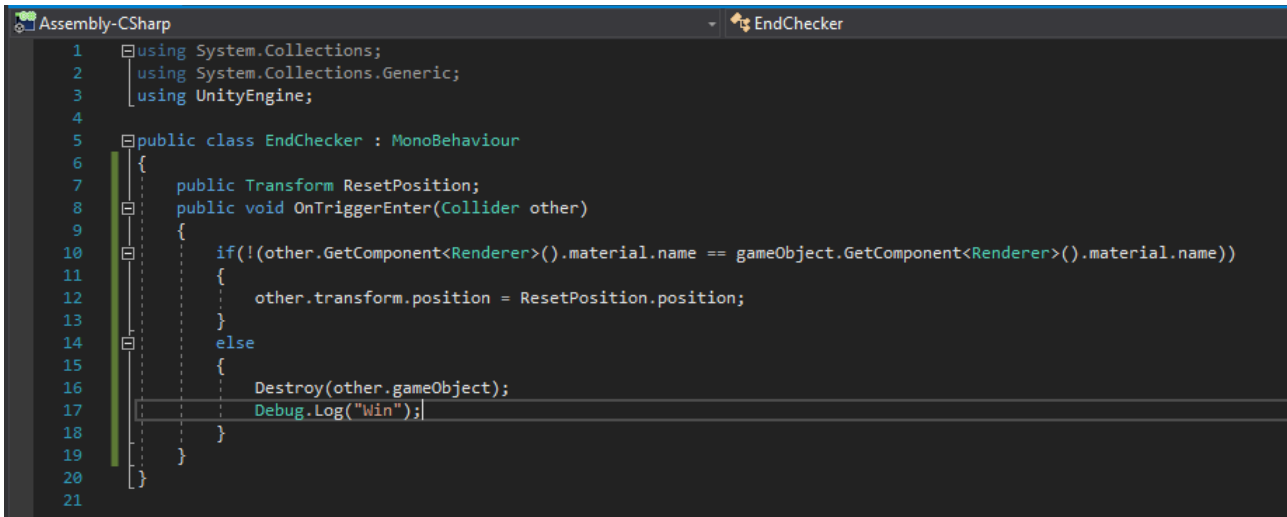
Konvertieren Sie dessen Collider auch zu einem Trigger.

Beispielresultat:



Erzeugen Sie ein neues Script, welches beim Eintritt in den Trigger des länglichen „Würfels“ (jetzt Quaders) die Materials des Quaders und der Kugel vergleicht (Hinweis: material.name vergleichen, weil es sich bei den Materials um Instanzen handelt).

Durch `Debug.Log(„Text“)`; kann schnell die Funktion geprüft werden.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class EndChecker : MonoBehaviour
6 {
7     public Transform ResetPosition;
8     public void OnTriggerEnter(Collider other)
9     {
10         if(!other.GetComponent<Renderer>().material.name == gameObject.GetComponent<Renderer>().material.name)
11         {
12             other.transform.position = ResetPosition.position;
13         }
14         else
15         {
16             Destroy(other.gameObject);
17             Debug.Log("Win");
18         }
19     }
20 }
21
```

Zusätzlich soll noch ein „Teleport“ durchgeführt werden. Dafür benötigen wir eine Referenzposition. Dazu kann man einfach ein „Empty GameObject“ erstellen und dieses an die Startposition der Kugel bewegen.

(Trick: Empty GameObject der Kugel in der Hierarchy unterordnen → position im Inspector auf 0/0/0 setzen → Empty GameObject aus der Unterordnung herausziehen)

Dieses „Empty GameObject“ dann noch als ResetPosition dem Checker Quader zuweisen und fertig ist das rudimentäre Spiel.

Teil 3: (Für besonders enthusiastische Teilnehmer / Teilnehmer mit zu viel Zeit)

Gestalten Sie das Spiel wiederholbar mit zufällig verteilten Materials.