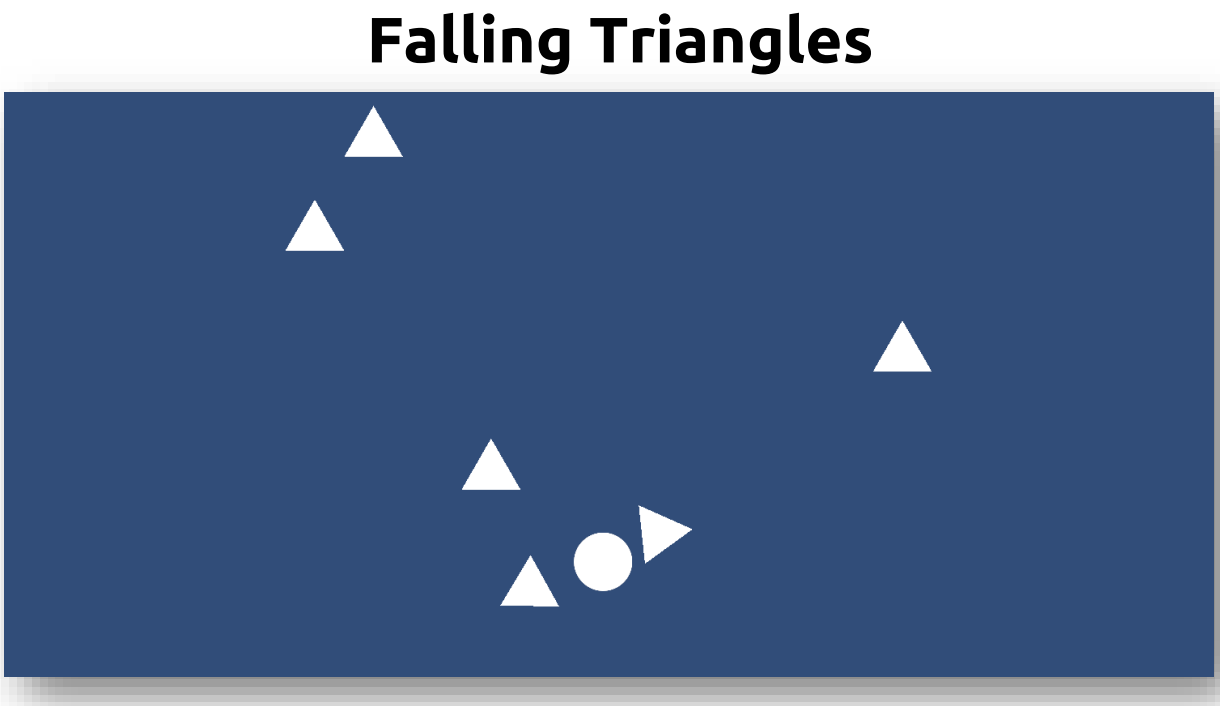
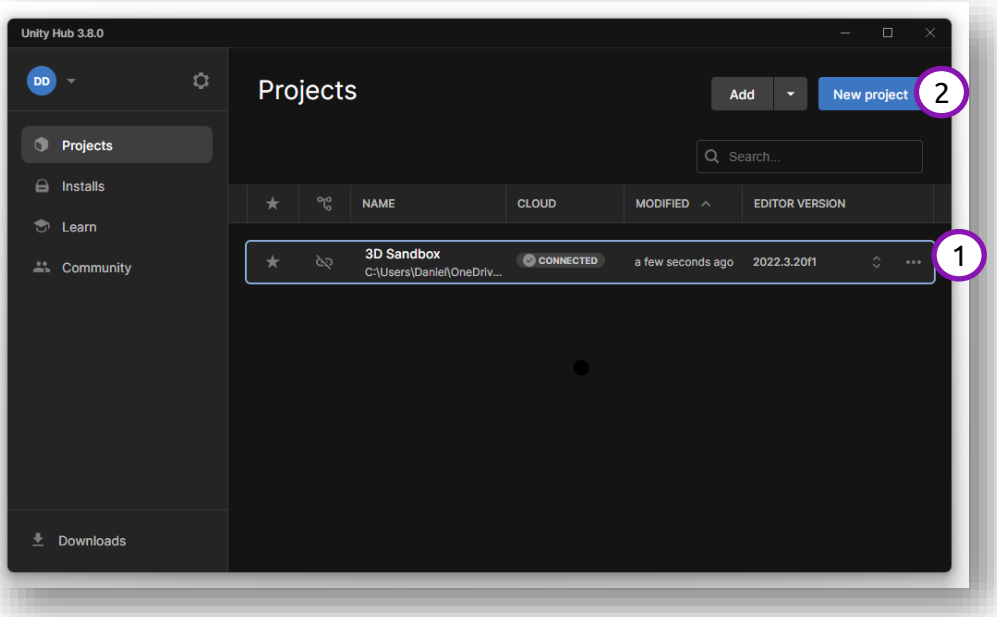
Unity – Erste Einheit

In dieser Einheit:



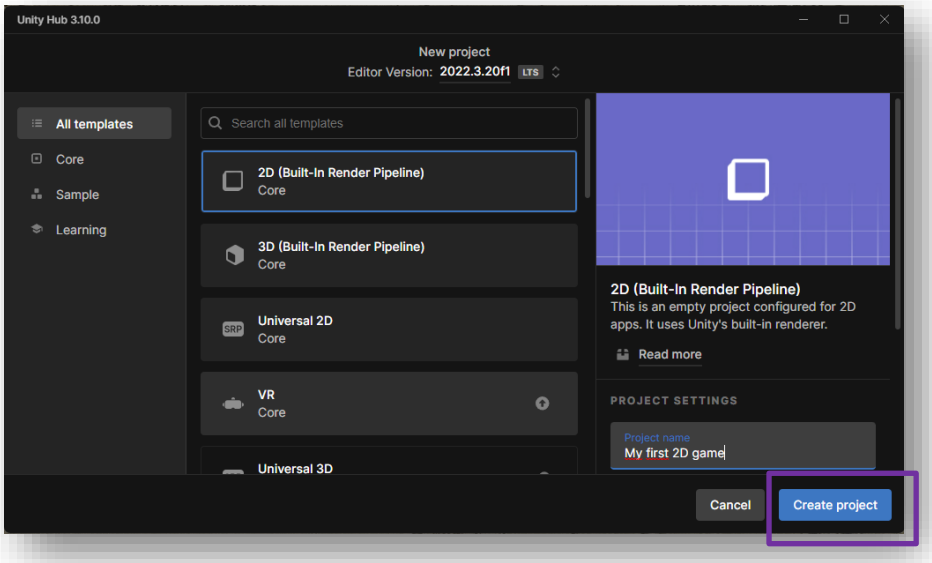
Inhalt

Der Unity Hub



Im Unity Hub findest du all deine Projekte.

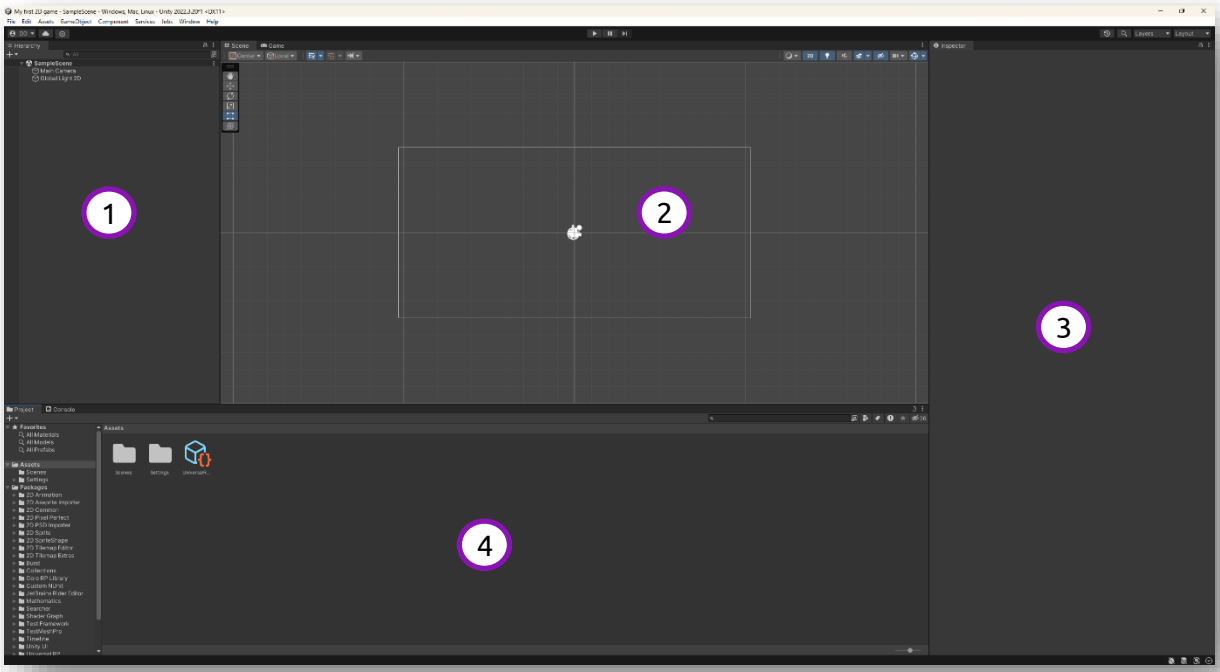
1. Hier ist dein aktuelles Projekt. Mit einem Klick kannst du es direkt öffnen.
2. Hier kannst du ein neues Projekt erstellen. **Klicke nun auf diesen Button**. Es sollte sich dieses Fenster öffnen:



Du kannst nun deine gewünschte Umgebung auswählen und deinem Projekt einen Namen geben. Für dieses Tutorial wähle bitte die „2D (Built-In Render Pipeline)“ Umgebung aus, **gib ihm einen Namen** *(hier: My first 2D game)* und drücke auf Create project.

Das User-Interface (UI) von Unity

Nachdem du dein Projekt erstellt und geöffnet hast, wirst du von einem (noch) leeren Unity UI begrüßt. Das sieht dann so aus:



1. Hierarchy: Hier siehst du all deine Objekte (inklusive der Kamera und dem Licht). Mit Rechtsklick in diesem Feld kannst du direkt neue Objekte erzeugen. Mit einem Klick auf ein Objekt wird dieses in der *Szene* **(2)** hervorgehoben und seine Eigenschaften im *Inspector* **(3)** angezeigt.

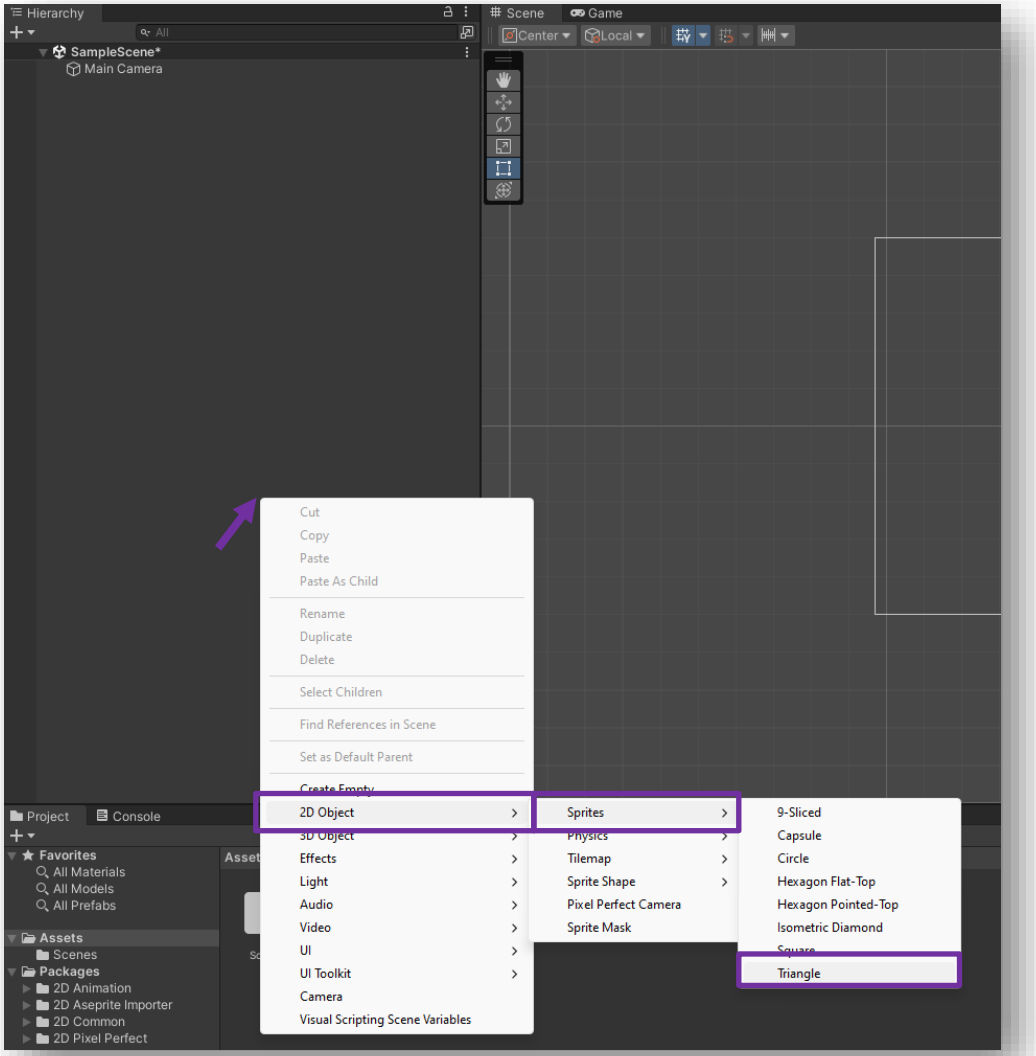
2. Scene: Hier werden deine Objekte visuell dargestellt. Jetzt gerade siehst du hier nur das Symbol für die Kamera und für die Beleuchtung.
   Mit der *rechten Maustaste* kannst dich umsehen.
   Mit dem *Mausrad* kannst du hinein- und hinauszoomen.
   Mit *drücken* vom *Mausrad* kannst du dich in der Szene seitlich bewegen.
   Mit der *linken Maustaste* kannst du, gleich wie in der *Hierarchy*, Objekte auswählen sodass du ihre Eigenschaften im *Inspector* angezeigt werden.

3. Inspector: Hier werden die Eigenschaften von Objekten angezeigt und können verändert werden. Weiters können hier Komponenten (Components) zu Objekten hinzugefügt werden.

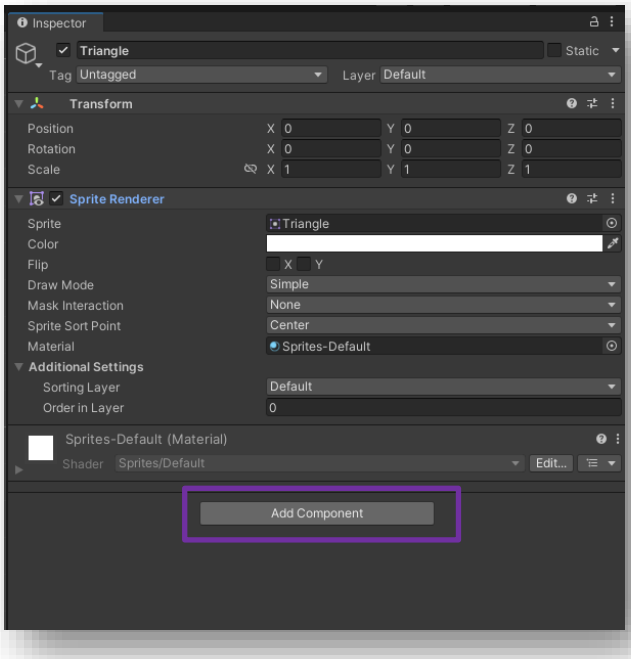
4. Project: In diesem Feld werden unsere (importierten) Modelle und Materialien (Assets) angezeigt. Von hier aus können wir diese per Drag-and-Drop in unsere Spielwelt einbringen.

Erstellen von Objekten - Dreieck

Als erstes erstellen wir nun in unserer Hierarchy ein Dreieck (**Triangle**). Dafür klicken wir mit der **rechten Maustaste** in Hierarchy und wählen dann **2D Object, Sprites, Triangle**.

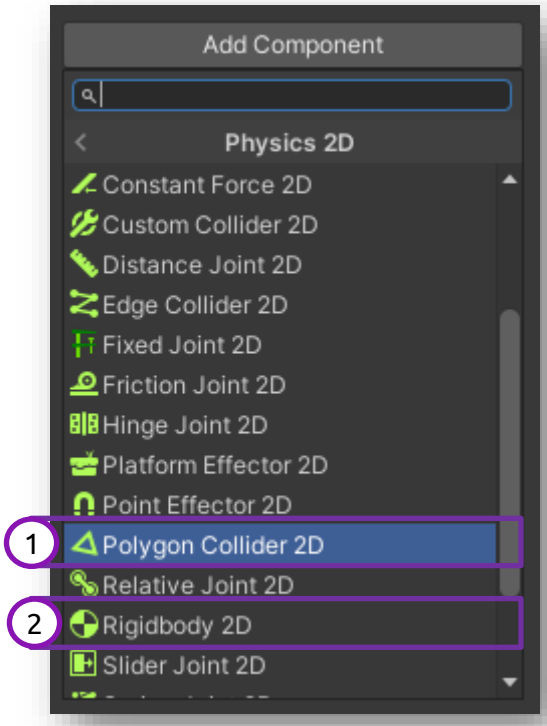


Dieses Objekt siehst du nun in Hierarchy. Wenn du es anklickst, siehst du dessen Eigenschaften im Inspector (siehe nächstes Bild).

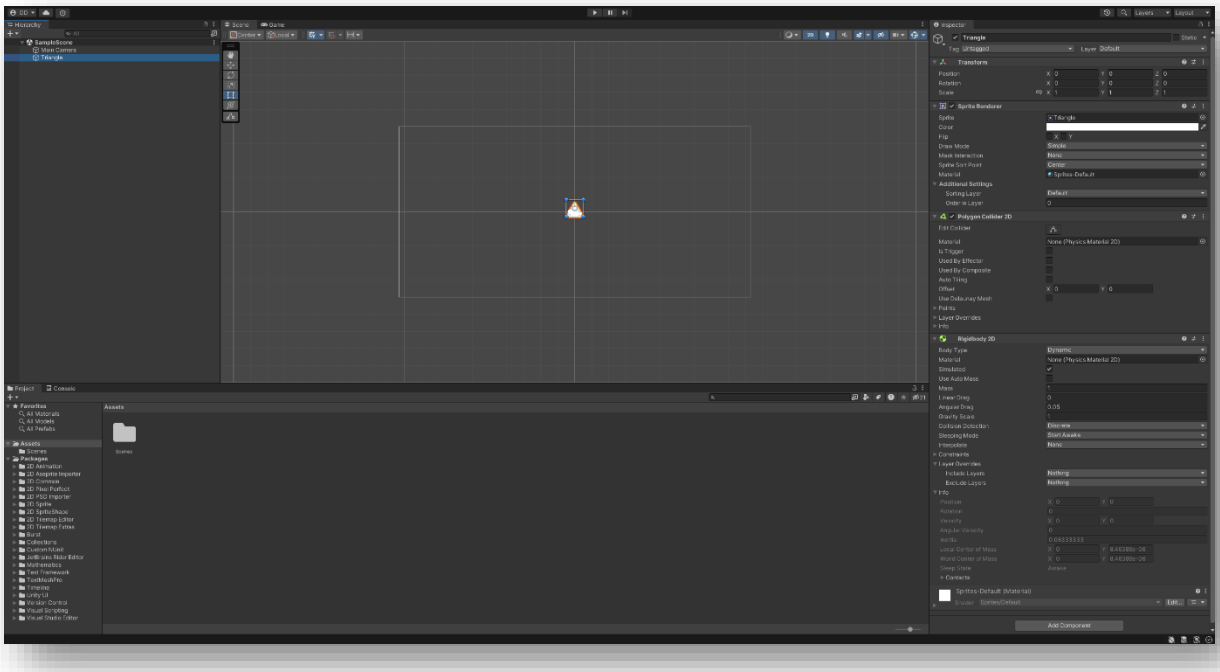


Als nächstes geben wir diesem Dreieck zwei neue Komponenten. Drücke dafür auf den Button unten (dieser heißt **Add Component)**.

Wir wählen unter **Physics 2D** den Polygon Collider 2D **(1)** und auch Rigidbody 2D **(2)** aus.

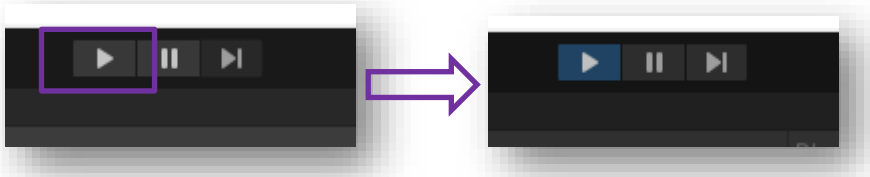


Wenn du alles richtig gemacht hast, sollte es (wenn du das Dreieck auswählst) so aussehen:



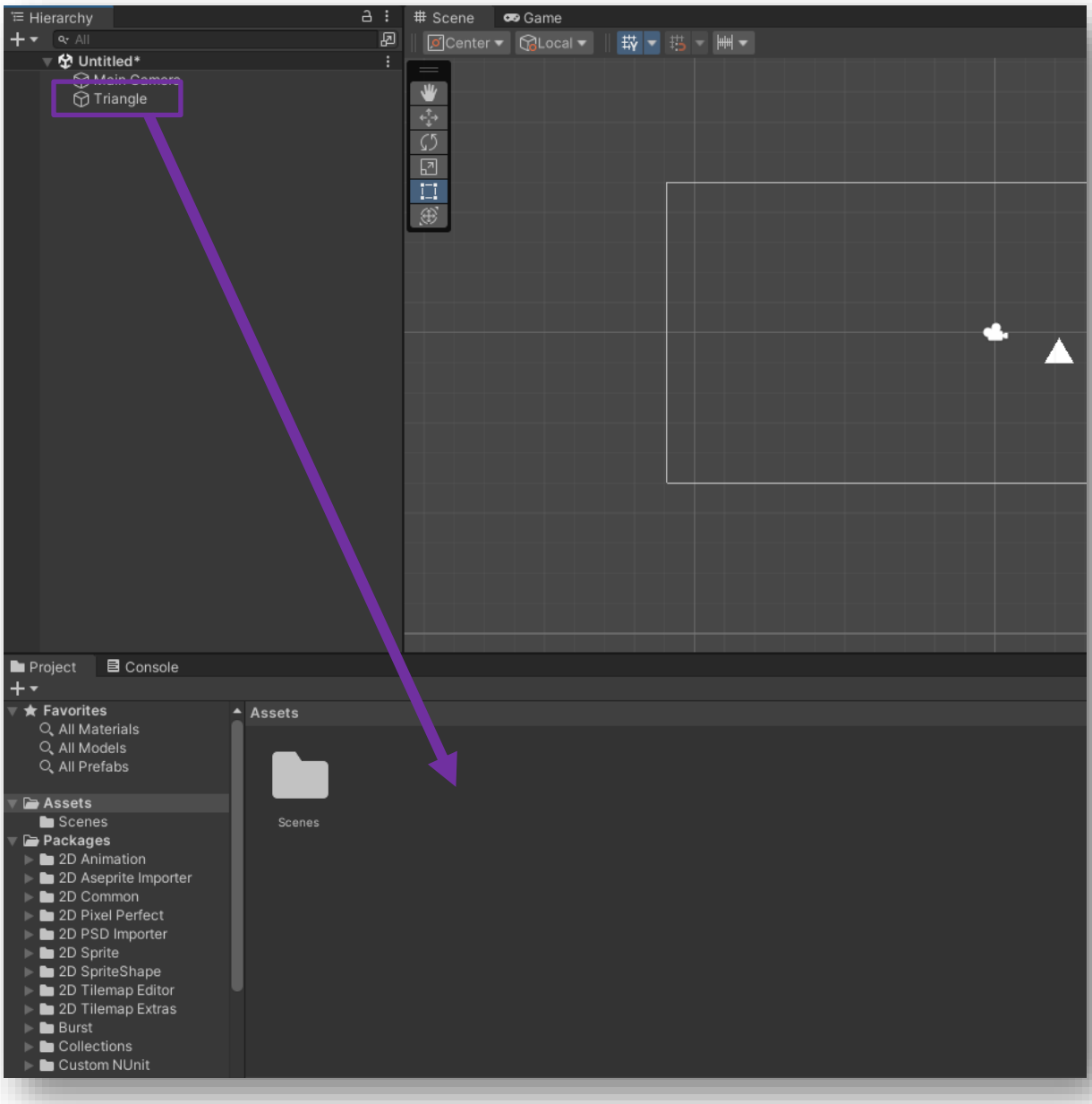
Du siehst in der Mitte über der Scene einen **Play-Button**.

Wenn du ihn drückst, startest du das Spiel und dein **Dreieck** sollte **hinunterfallen**. Während das Spiel läuft, wird der Button blau. Drückst du ihn nochmal, wird das Spiel beendet.



Der Assets Ordner

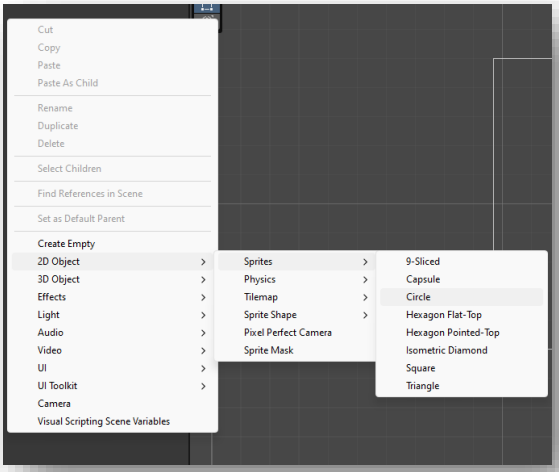
Jetzt ziehen wir unser **Triangle** von der Hierarchy in Project (in den Assets Ordner, welcher automatisch geöffnet sein sollte). Stelle sicher, dass das Objekt auch wirklich in Project ist. Nun löschen wir **Triangle** aus der Hierarchy. Klicke es dafür in der Hierarchy an und drücke die ENTFERNEN-Taste (**Entf** am Keyboard). Wir möchten nämlich nicht, dass das **Triangle** beim Start des Spiels zu sehen ist.



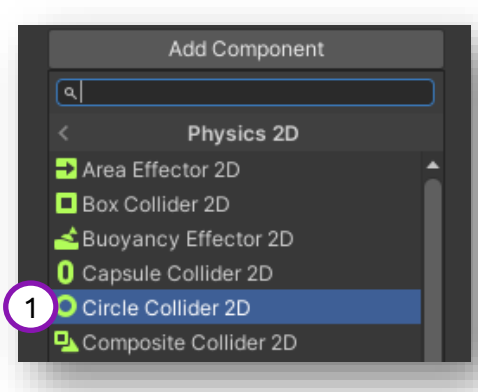
Erstellen von Objekten - Kreis

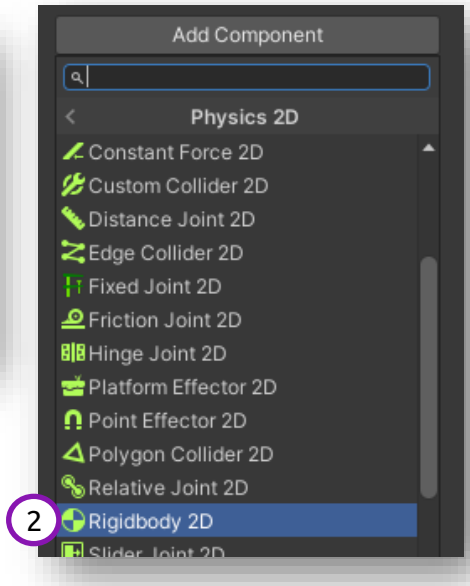
Nachdem wir unser Dreieck erstellt, mit Komponenten erweitert und in das Project gebracht haben, erstellen wir nun unseren Spieler. Dieser ist in diesem Spiel ein **Kreis** (Circle).

Dafür klicken wir mit der **rechten Maustaste** in die Hierarchy, und wählen aus 2D Object, Sprites, Circle

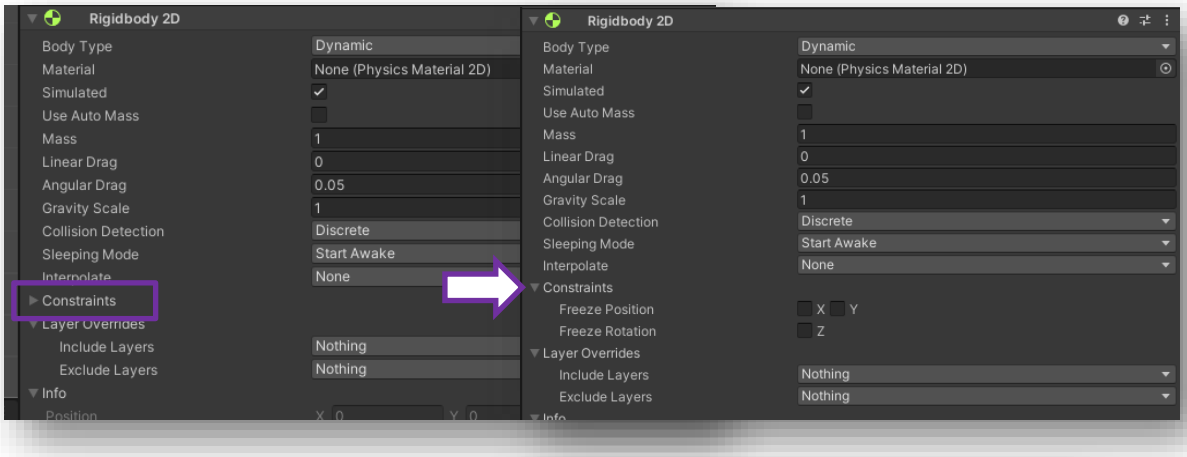


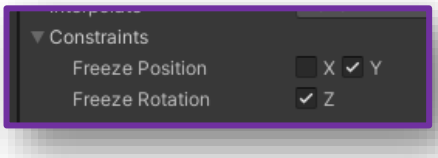
Diesem **Circle** (Kreis) geben wir nun wieder die zwei Komponenten. Dafür klicken auf den **Cirlce**, und im Inspector **Add Component**, danach wählen wir unter **Physics 2D** den Circle Collider 2D **(1)** und auch Rigidbody 2D **(2)** aus.

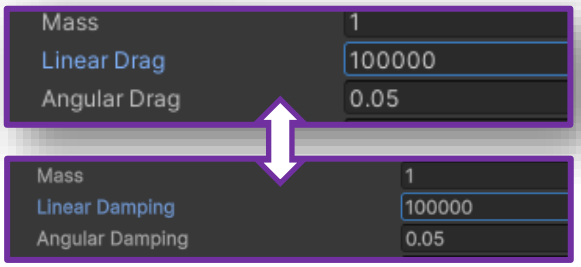




Sehen wir uns nun im Inspector von **Circle** die Komponente **Rigidbody 2D** an. Hier gibt es eine Eigenschaft die **Constraints** heißt. Klicke auf sie, um sie auszuklappen.







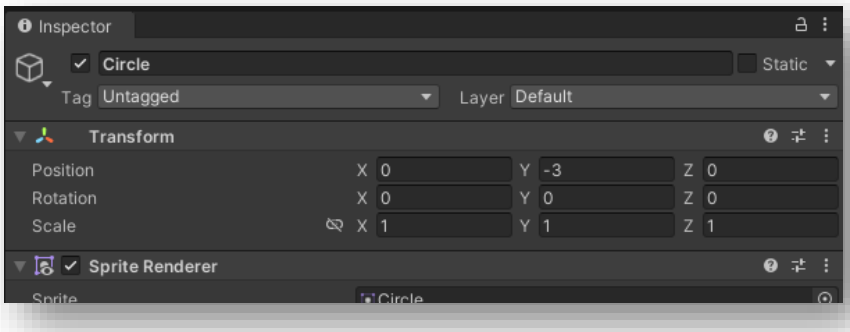
In Unity 6 heißt die Einstellung Linear Damping

Wähle nun **Freeze Position „Y“** und **Freeze Rotation „Z“** aus und setze ein Häkchen.

Zum Schluss ändern wir noch den Wert bei **Linear Drag**, welchen wir etwas darüber finden auf *100000*.

Was machen diese Werte? Freeze Position und Freeze Rotation verhindert, dass unser Ball, nachdem er mit etwas zusammengestoßen ist nach unten verschoben wird bzw. anfängt sich zu drehen.

Jetzt ändern wir noch die Position von unserem **Circle**. Das tun wir, indem wir ihn anklicken und im Inspector den Wert „-3“ beim **Position Y** eingeben. Die anderen Werte lassen wir bei ihren anfänglichen Werten bzw. setzen sie auf „**0**“.

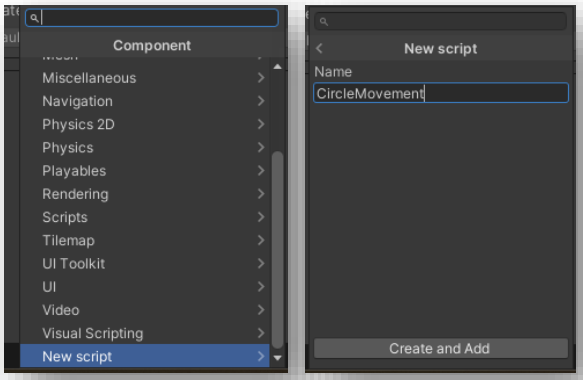


Erstellen von Skripten - Kreis

Nun geben wir unserem **Circle** noch eine Komponente. Diesmal wird es ein bisschen komplizierter; wir schreiben unser erstes Programm! Aber keine Sorge, der Code, den du dafür benötigst steht weiter unten und kann problemlos verwendet werden.

Also wählen wir unseren **Circle** aus, klicken im Inspector auf **Add Component** und wählen **New Script** aus.

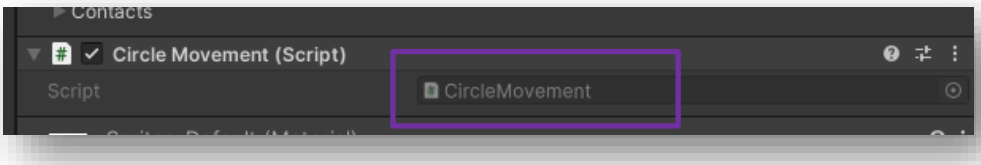
Dieses nennen wir „**CircleMovement**“ und klicken auf **Create and Add**.



Achte unbedingt darauf, dass das Skript auch wirklich **CircleMovement** heißt und richtig geschrieben ist.

Du kannst das Skript nun über zwei Wege öffnen;

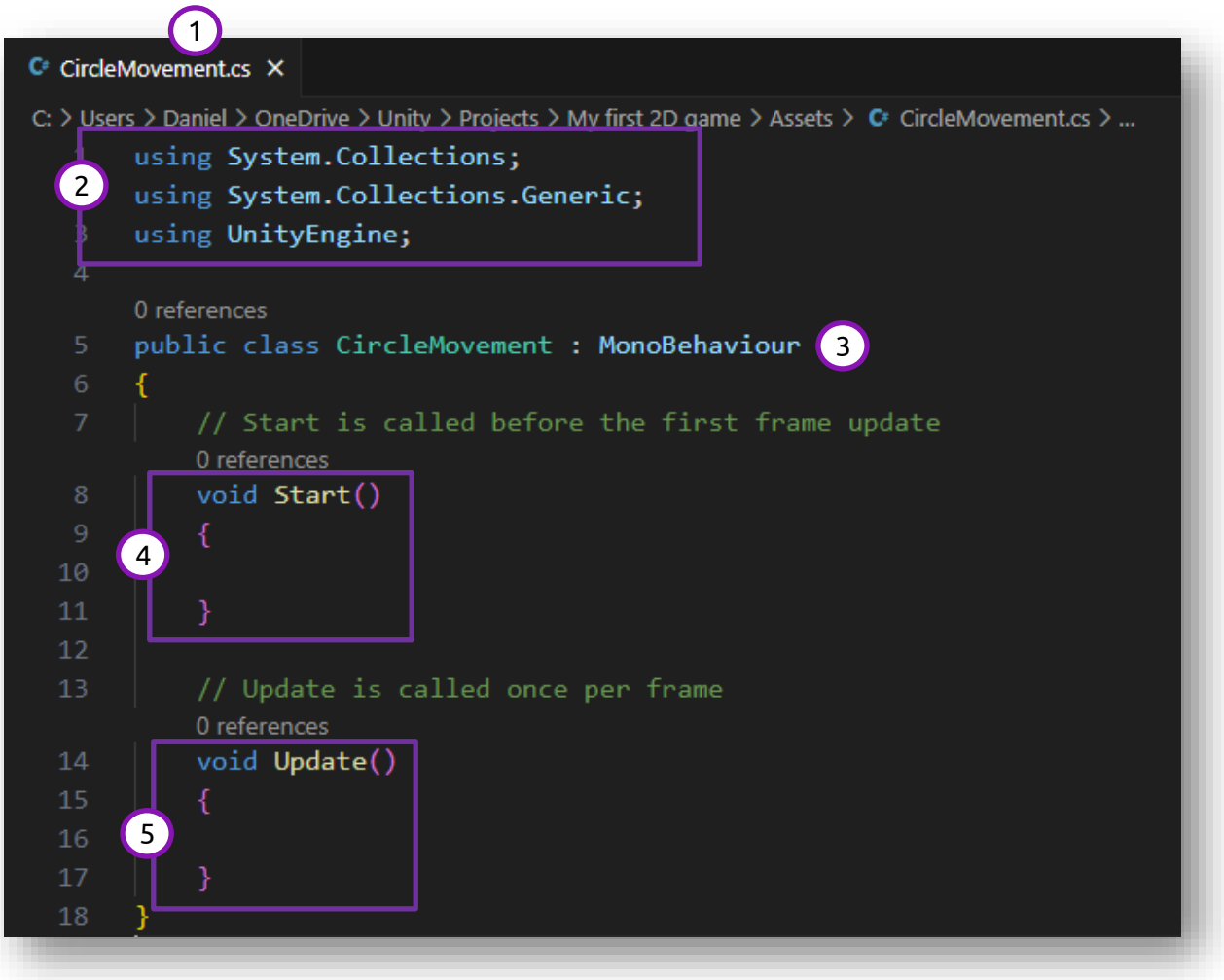
1. Du wählst unseren **Circle** aus, und in dessen **Komponenten** (im Inspector) findest du das Skript. Mit einem **Doppelklick** darauf kannst du es öffnen.



oder

2. Du doppelklickst in Projects direkt auf das Skript **CircleMovement**.

Nun öffnet sich ein neues Fenster. Hier siehst du das noch leere Programm.



1. Der Name unserer Klasse (und unseres Skripts)
2. Hier wird alles aufgelistet das für diese Klasse importiert wird. Wir können diese Liste natürlich erweitern und verändern
3. Der Name der Klasse, dieser sollte gleich sein wie (1), ignoriere das „ *: MonoBehaviour*“.
4. Die **Start**-Methode wird aufgerufen, wenn das Programm das erste Mal gestartet wird
5. Die **Update**-Methode wird jeden Frame aufgerufen. D.h. hier werden Berechnungen durchgeführt.

Wir **ersetzen** jetzt allerdings **alles,** was in diesem Skript steht mit dem folgenden:

(**Wähle alles** in der **Textbox** aus, drücke „**Strg + C**“, **öffne** dein Programmfenster (das **Skript**), drücke „**Strg + A**“ (damit wählst du alles vom Skript aus) und „**Strg + V**“ um den kopierten Code einzufügen)

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

public class CircleMovement : MonoBehaviour
{

  public float movementSpeed = 25f;

  private float minX = -10f;
  private float maxX = 10f;

  // Start is called before the first frame update
  void Start()
  {
    Application.targetFrameRate = 60;
  }

  // Update is called once per frame
  void Update()
  {
    // Store the current position in a temporary variable
    Vector3 newPosition = transform.position;

    // Update the X position based on input
    newPosition.x += Input.GetAxis("Horizontal") * movementSpeed * Time.deltaTime;

    // Clamp the X position
    newPosition.x = Mathf.Clamp(newPosition.x, minX, maxX);

    // Apply the clamped position back to the transform
    transform.position = newPosition;
  }
}
```

MES_CircleMovement.txt

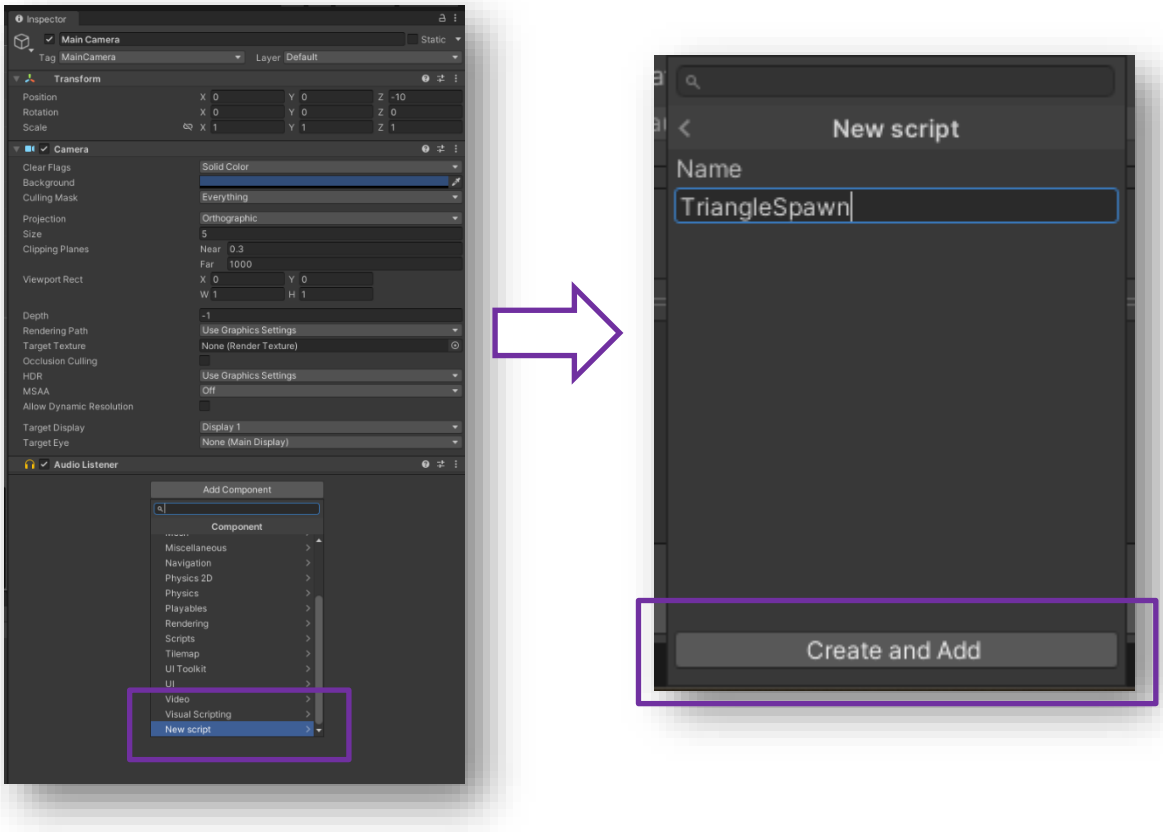
Nachdem du alles reinkopiert hast, **speichere** das Skript mit „**Strg + S**“.

Wenn du nun das **Spiel startest** (über der Scene auf den **Play-Button**), kannst du den **Circle bewegen**. Dir sollte nun auffallen, dass du dich nur innerhalb des Bildschirms und nur nach **links** und **rechts** bewegen kannst.

Erstellen von Skripten - Logik

Erinnerst du dich noch an das **Triangle** (Dreieck) das wir früher erstellt haben? In diesem Kapitel lernst du, wie wir **Dreiecke spawnen** (erscheinen) lassen können.

Wir wählen nun die **Main Camera** in Hierarchy aus und fügen (im Inspector) ein **neues Skript** (**Add Component** - **new Script**) hinzu. Dieses nennen wir „**TriangleSpawn**“. *Achte auch hier wieder, dass du es genauso geschrieben hast.*



Öffne das **Skript** und **kopiere**, gleich wie beim Circle, den **Code** in das Programm.

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

public class TriangleSpawn : MonoBehaviour
{

    public float delay = 0.3f;
    public GameObject triangle;

    // Start is called before the first frame update
    void Start()
    {
        InvokeRepeating("Spawn", delay, delay);
    }

    // Update is called once per frame
    void Update()
    {

    }

    void Spawn() {
        Destroy(Instantiate(triangle, new Vector3(Random.Range(-6,6), 10, 0), Quaternion.identity),3);
    }

}
```

MES_TriangleSpawn.txt

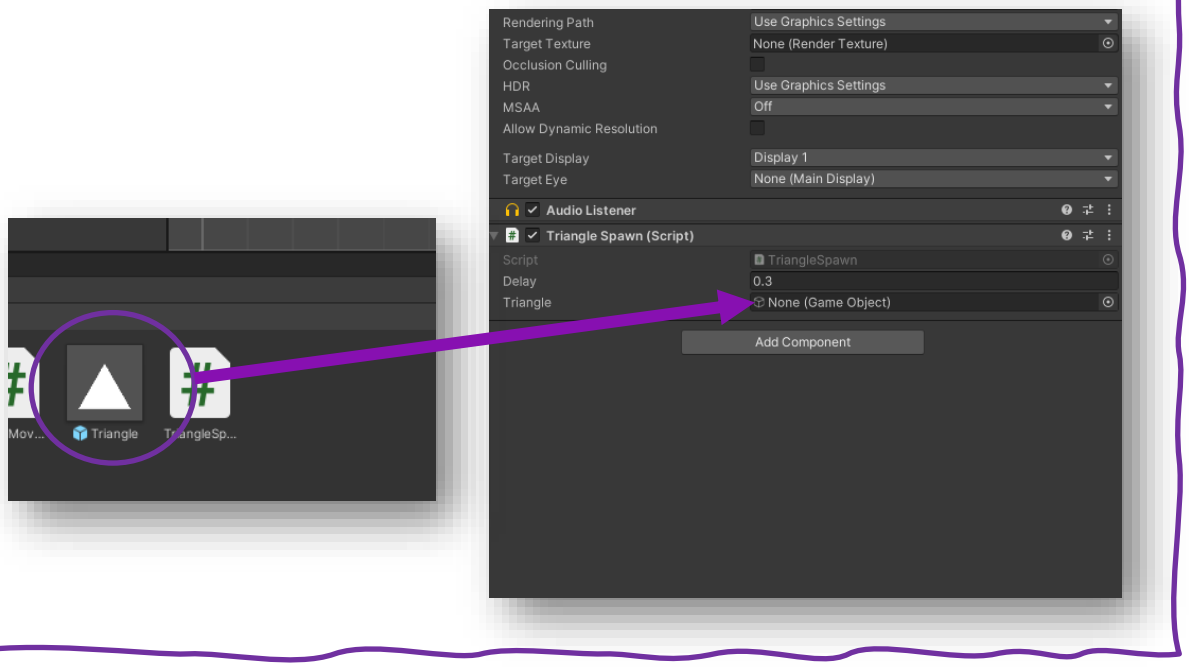
Nachdem du alles reinkopiert hast, **speichere** das Skript mit „**Strg + S**“.

Wenn du nun das **Spiel startest**, wirst du sehen, dass sich **nichts verändert** hat. Aber warum nur?

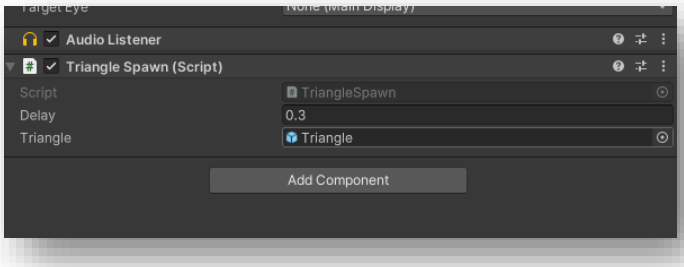
Wähle in Hierarchy wieder **Main Camera** aus und suche unser **TriangleSpawn** Script. Du siehst hier nun zwei Variablen – **Delay** und **Triangle**.

- **Delay** ist mit 0.3 eingestellt
- Bei **Triangle** steht „None (Game Object)“.

Wir ziehen nun unser **Triangle** von Project in dieses Feld.



Danach müsste es so aussehen:



Starte nun das Spiel; du siehst, dass **Triangles** außerhalb des Bildschirms spawnen und runterfallen. Versuche nun jedem Objekt auszuweichen.

Was passiert wenn du bei **Circle** im Inspector bei der Komponente..

- **Rigidbody 2D** die Variable **Linear Drag/Damping** von 100000 auf 1 änderst?
- **Circle Movement (Script)** den Movement Speed veränderst?

Bleiben die neuen Werte gespeichert, wenn du das Spiel beendest?

Danke, dass du diese Anleitung für die 2D-Spieleentwicklung in Unity durchgemacht hast.