Einstieg in Unity Teil 2 – Der drehende Würfel

Description

Nachdem wir uns im ersten Teil in Unity umgeschaut und eine kleine Szene gebastelt haben, geht es nun an die Programmierung. Unser Ziel ist es, den Würfel in alle drei Richtungen zu drehen.

Wieso das denn?

Wir schauen uns an, wie man Skripte erstellt, Objekten zuweist und Eigenschaften – in diesem Fall die Rotation – manipuliert. Außerdem stellen wir dabei die Eigenschaften dem Objekt so zur Verfügung, dass wir sie im Inspektor verändern können.

Das Drehen und Bewegen von Objekten ist in der Spieleentwicklung ohnehin sehr wichtig. Stellen wir uns einen Deckenventilator vor, oder eine Waffe, die sich in Richtung des Spielers dreht. Alles, was in diesem Teil gezeigt wird, ist für die Entwicklung von Spielen elementar.

Skript erstellen

Das ist denkbar einfach: Klicke mit der linken Maustaste in die Assets (Ressourcen) in einen freien Bereich. Dann wähle im Menü *Create* ? *C# Script*. Anschließend gibst Du dem Skript einen Namen. Bei mir heißt es *Object_Rotation*.

Fertig, du hast dein erstes Skript erstellt. Doch wir wollen mehr, also doppelklicke es. Es sollte sich nun ein Editor öffnen, vorzugsweise *Visual Studio Code*.

Was uns Unity liefert

Wenn das Skript im Editor geladen wurde, wirst du feststellen, dass da bereits einiges drin steht.

```
using UnityEngine;

public class Object_Rotation : MonoBehaviour
```

```
{
    public Vector3 rotationSpeeds = new Vector3(30.0f, 15.0f, 45.0f); // Gesch

    void Start()
    {

    }

    void Update()
    {
        // Holen dir die aktuelle Rotation des Objekts als Quaternion
        Quaternion currentRotation = transform.rotation;

        // Erzeuge Quaternion-Deltas für die Rotation in den X, Y und Z-Achsen
        Quaternion deltaRotationX = Quaternion.Euler(rotationSpeeds.x * Time.d
        Quaternion deltaRotationY = Quaternion.Euler(0, rotationSpeeds.y * Tim
        Quaternion deltaRotationZ = Quaternion.Euler(0, 0, rotationSpeeds.z *

        // Multipliziere die aktuellen Rotationen mit den Delta-Rotationen
        currentRotation *= deltaRotationX;
        currentRotation *= deltaRotationY;
        currentRotation *= deltaRotationZ;

        // Setze die neue Rotation
        transform.rotation = currentRotation;
    }
}
```

Gehen wir den Code schrittweise durch.

`public Vector3 rotationSpeeds = new Vector3(30.0f, 15.0f, 45.0f);`: Hier wird eine öffentliche Variable `rotationSpeeds` deklariert und initialisiert. Diese Variable ist vom Typ `Vector3` und enthält Geschwindigkeiten für die Rotation um die X-, Y- und Z-Achsen. Das wirst du später auch im Inspektor verändern können.

`void Update() { }`: Die `Update()`-Methode wird einmal pro Frame aufgerufen. Hier wird die eigentliche Logik für die Rotation implementiert.

`Quaternion currentRotation = transform.rotation;`: Hier wird die aktuelle Rotation des Spielobjekts als Quaternion in der Variable `currentRotation` gespeichert.

Ein **Quaternion** ist eine mathematische Struktur, die in der 3D-Computergrafik und der Robotik verwendet wird, um Rotationen zu repräsentieren. Im Kontext von Unity und vielen anderen 3D-Grafik-Engines wird der Begriff häufig verwendet.

Im Vergleich zu den gebräuchlicheren Euler-Winkeln (Roll, Pitch, Yaw) haben Quaternionen einige Vorteile bei der Darstellung von Rotationen, insbesondere wenn es um das sogenannte „Gimbal Lock“ geht, das bei Euler-Winkeln auftreten kann.

Quaternion-Deltas für die Rotation in den X, Y und Z-Achsen erstellen:

- `Quaternion deltaRotationX = Quaternion.Euler(rotationSpeeds.x *`

```
Time.deltaTime, 0, 0);
```
- `Quaternion deltaRotationY = Quaternion.Euler(0, rotationSpeeds.y * Time.deltaTime, 0);`
- `Quaternion deltaRotationZ = Quaternion.Euler(0, 0, rotationSpeeds.z * Time.deltaTime);`

Aktuelle Rotation mit den Delta-Rotationen multiplizieren:

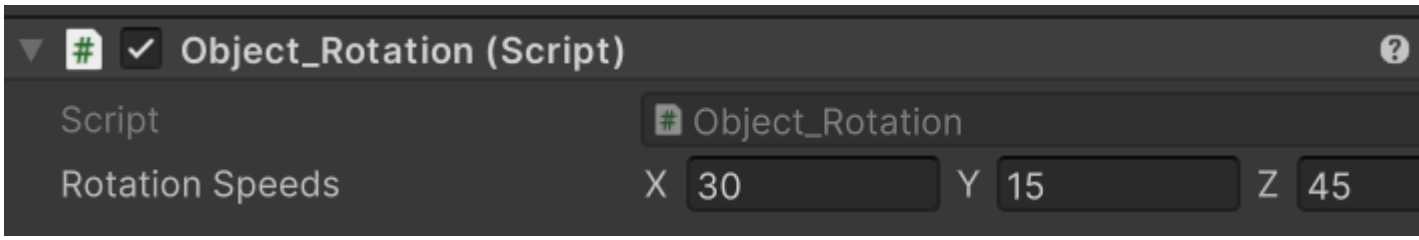
-
  - `currentRotation *= deltaRotationX;`
  - `currentRotation *= deltaRotationY;`
  - `currentRotation *= deltaRotationZ;`
- **`transform.rotation = currentRotation;`**: Schließlich wird die neue Rotation des Spielobjekts auf Basis der berechneten Änderungen gesetzt.

Somit dreht sich das Spielobjekt pro Frame um die angegebenen Rotationsgeschwindigkeiten um die X, Y und Z-Achsen. Die Rotationsgeschwindigkeiten werden mit `Time.deltaTime` multipliziert, um sicherzustellen, dass die Rotation in Abhängigkeit von der vergangenen Zeit seit dem letzten Frame erfolgt, und somit unabhängig von der Framerate des Spiels ist.

Skript zuweisen und starten

Wenn das Skript gespeichert wurde, wechselst du in Unity. Da wird es sofort überprüft. Sollten Fehler auftauchen, stehen diese in der Konsole. Das Spiel kann erst gestartet werden, wenn es keine Fehler mehr im Skript gibt.

Ist das Skript fehlerfrei, kannst du es dem Würfel zuweisen. Klicke dafür den Würfel an. Du siehst seine Eigenschaften im Inspektor. Nun wähle das Skript bei den Ressourcen aus, halte die Maustaste gedrückt und ziehe es in den Inspektor ganz unten hin.



Objekt Rotation im Inspektor

Du siehst, dass Unity die drei Geschwindigkeiten anzeigt. Du kannst die Werte ändern, sogar, wenn die Vorschau gestartet wurde.

Die Vorschau startest du, indem du über der 3D-Szene auf den **Play-Button** drückst. Wenn du später noch einmal drauf drückst, beendest du die Vorschau.

Nun sollte sich der Würfel in drei Richtungen drehen. Wie gesagt, kannst du die Geschwindigkeiten der drei Achsen jetzt weiter ändern, bis es für dich optimal ist. Du kannst auch mit negativen Zahlen arbeiten.

Dieses Skript kannst du später allen anderen Objekten zuweisen und wird für dich immer wieder von Nutzen sein.

Nächster Teil

Das war es auch schon wieder. Im dritten und letzten Teil dieser kleinen Einstiegsserie werden wir weiter mit Code arbeiten. Unser Ziel wird darin bestehen, per Code aus einem einzigen Würfel viele zu erschaffen, die sich wie ein gewaltiger Würfel drehen.

Weiterführende Links

Einstieg in Unity Teil 1 – 3D und erste Szene
Einstieg in Unity Teil 3 – Die Macht der Programmierung
Warum soll ich programmieren lernen?
Arcade-Spiele – die Mutter aller Genres?
Frustspirale – Designschnitzer in Spieleklassikern
Mehr als nur eine Party

Externe Links

Unity Dokumentation
Instanziierung – Wikipedia
Quaternion – Wikipedia
Eulersche Zahl – Wikipedia
Eulerscher Winkel – Wikipedia
Gimbal Lock – Wikipedia

Date Created
1. Dezember 2023
Author
sven