ToiletPaper #139

Unity – Persistente Objekte zwischen Szenen leicht gemacht

Autor: Christian Gerhards / Software Engineer / Business Division Automotive Bavaria

✗ Problem

In Unity gibt es mehrere Möglichkeiten, Daten oder Objekte zwischen dem Laden von Szenen persistent zu halten – z. B. DontDestroyOnLoad, Singletons, ScriptableObjects. Manche sind besser, manche schlechter, aber wirklich ideal sind sie nicht. Es gibt aber einen cleveren Weg, auf den ich während meines privaten Spieleprojekts gestoßen bin – persistente Objekte in einer persistenten Szene halten.

✓ Lösung

Unity kann schon länger multiple Szenen gleichzeitig verwalten. Diese Funktion eignet sich gut für das dynamische Laden und Entladen von Szeneteilen, z. B. Chunks in einer Voxel-ähnlichen Umgebung, einzelnen Map-Teilen usw., aber auch für die Objektverwaltung. Um Objekte persistent zu halten, werden diese in eine Szene gelegt, die während der Laufzeit immer geladen bleibt. Diese persistente Szene ist die einzige Szene, die beim Spielstart bereits geladen ist und entscheidet, welche weiteren Szenen geladen und entladen werden sollen. Dies ist durch den additiven Lademodus des SceneManagers möglich.

➔ Beispiel

Stell dir vor, es soll immer nur eine persistente Szene und eine Map-Szene im Spiel aktiv sein. Map-Szenen sind der Teil, den wir dynamisch laden und entladen wollen und der z. B. eine Tilemap eines Levels enthält.

SceneLoadController.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneLoadController : MonoBehaviour{
    private string nameOfSceneToLoad, nameOfCurrentScene;
    public string nameOfFirstMapScene;

    // Load first map scene, persistent scene is already loaded
    void Start(){LoadMapScene(nameOfFirstMapScene);}

    // Set callback for loading new map scene, start unloading
    public void LoadNewMapScene(string nameOfSceneToLoad){
        SceneManager.sceneUnloaded += MapSceneUnloadFinished;
        this.nameOfSceneToLoad = nameOfSceneToLoad;
        SceneManager.UnloadSceneAsync(nameOfCurrentScene);
    }

    // Remove callback, call load for new map scene
    private void MapSceneUnloadFinished(Scene unloadedScene){
        SceneManager.sceneUnloaded -= MapSceneUnloadFinished;
        if (!SceneManager.GetSceneByName(nameOfSceneToLoad).isLoaded){
            LoadMapScene(nameOfSceneToLoad);
        }
    }

    // Set currentSceneName, load a map scene in additive mode to keep persistent scene
    private void LoadMapScene(string nameOfSceneToLoad){
        nameOfCurrentScene = nameOfSceneToLoad;
        SceneManager.LoadScene(nameOfSceneToLoad, LoadSceneMode.Additive);
    }
}
```

✚ Weiterführende Aspekte

- Unity API für den SceneManager
- Catlike Coding – Objekt-Management durch mehrere Szenen