ToiletPaper #139

Unity – Persistent objects between scenes in a simple way

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

✗ Problem

Unity has multiple ways to keep data or objects persistent between scene loads. E.g. DontDestroyOnLoad, Singletons, ScriptableObjects. Some are better, some are worse and they are all not ideal. But there is a really smarter path I stumbled upon during my private gaming project. Keep persistent objects in a persistent scene.

✓ Solution

Unity has been able to handle multiple scenes at once for a long time. This is great for dynamic loading and unloading of scene parts e.g. chunks in a voxel like environment, single map-parts etc. but also for your object management. To keep your objects persistent, put them in a scene which you won't unload at runtime and is always loaded. This persistent scene is the only scene which is already loaded at game start and decides which additional scenes should be loaded and unloaded. This is possible due to the SceneManager's additive loading mode.

➔ Example

Imagine we want to have always only one persistent scene and one map scene active in our game. Map scenes are the part we want to load and unload dynamically and containing for example a tilemap of a level.

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);
    }
}
```

✚ Further Aspects

- Unity Api for the SceneManager
- Catlike Coding - Object Management via Multiple Scenes