﻿//This is a modification of the original code found here: http://forum.unity3d.com/threads/simple-reusable-object-pool-help-limit-your-instantiations.76851/

//Remember kids, credit where credit is due!

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

public class ObjectPool : MonoBehaviour

{

public static ObjectPool current; //A public static reference to itself (make's it visible to other objects without a reference)

public GameObject[] prefabs; //Collection of prefabs to be poooled

public List<GameObject>[] pooledObjects; //The actual collection of pooled objects

public int[] amountToBuffer; //The amount to pool of each object. This is optional

public int defaultBufferAmount = 10; //Default pooled amount if no amount abaove is supplied

public bool canGrow = true; //Whether or not the pool can grow. Should be off for final builds

GameObject containerObject; //A parent object for pooled objects to be nested under. Keeps the hierarchy clean

void Awake ()

{

//Ensure that there is only one object pool

if (current == null)

current = this;

else

Destroy(gameObject);

//Create new container

containerObject = new GameObject("ObjectPool");

//Create new list for objects

pooledObjects = new List<GameObject>[prefabs.Length];

int index = 0;

//For each prefab to be pooled...

foreach ( GameObject objectPrefab in prefabs )

{

//...create a new array for the objects then...

pooledObjects[index] = new List<GameObject>();

//...determine the amount to be created then...

int bufferAmount;

if(index < amountToBuffer.Length)

bufferAmount = amountToBuffer[index];

else

bufferAmount = defaultBufferAmount;

//...loop the correct number of times and in each iteration...

for ( int i = 0; i < bufferAmount; i++)

{

//...create the object...

GameObject obj = (GameObject)Instantiate(objectPrefab);

//...give it a name...

obj.name = objectPrefab.name;

//...and add it to the pool.

PoolObject(obj);

}

//Go to the next prefab in the collection

index++;

}

}

public GameObject GetObject( GameObject objectType)

{

//Loop through the collection of prefabs...

for(int i=0; i<prefabs.Length; i++)

{

//...to find the one of the correct type

GameObject prefab = prefabs[i];

if(prefab.name == objectType.name)

{

//If there are any left in the pool...

if(pooledObjects[i].Count > 0)

{

//...get it...

GameObject pooledObject = pooledObjects[i][0];

//...remove it from the pool...

pooledObjects[i].RemoveAt(0);

//...remove its parent...

pooledObject.transform.parent = null;

//...and return it

return pooledObject;

}

//Otherwise, if the pool is allowed to grow...

else if(canGrow)

{

//...write it to the log (so we know to adjust our values...

Debug.Log("pool grew when requesting: " + objectType + ". consider expanding default size.");

//...create a new one...

GameObject obj = Instantiate(prefabs[i]) as GameObject;

//...give it a name...

obj.name = prefabs[i].name;

//...and return it.

return obj;

}

//If we found the correct collection but it is empty and can't grow, break out of the loop

break;

}

}

return null;

}

public void PoolObject ( GameObject obj )

{

//Find the correct pool for the object to go in to

for ( int i=0; i<prefabs.Length; i++)

{

if(prefabs[i].name == obj.name)

{

//Deactivate it...

obj.SetActive(false);

//..parent it to the container...

obj.transform.parent = containerObject.transform;

//...and add it back to the pool

pooledObjects[i].Add(obj);

return;

}

}

}

}