Spawner Service

Introduction

The TilePlus system uses pooling as much as possible and reasonable. Most of this activity is behind the scenes, but there is one pooler available for general use as a Service: SpawnerService.

SpawnerService is based on the TpSpawner : ScriptableRuntimeService<TpSpawner> class.

Use

SpawnerService can be used for painting tiles, which is a specialized use primarily for TpAnimZoneSpawner. In that case there’s no object pooling, which applies only to Prefabs. That use isn't further discussed here.

A tile can use SpawnerService to spawn pooled prefabs, with preloading. That feature is used by TpAnimZoneSpawner and TpAnimatedSpawner. You can examine those tiles' code to see how it works.

You can use SpawnerService from any code, not just tiles. Just use the SpawnPrefab method shown below:

```
public GameObject? SpawnPrefab(GameObject? prefab,
                                    Vector3     position,
                                    Transform?  parentTransform = null,
                                    string      parentNameOrTag = "",
                                    bool        searchForTag = false,
                                    bool        keepWorldPosition = true)
```

The first two parameters are obvious.

parentTransform: If provided then the spawned Prefab is parented to that transform. If the value is null a search is done for the parent using the parentNameOrTag string. If that string has a value, then one of two things occurs:

- searchForTag = true: Use GameObject.FindWithTag to locate a GameObject to use for a parent.

- searchForTag = false; Use GameObject.Find to locate a GameObject to use for a parent (slower).

If neither search provides a parent, or if parentNameOrTag is null or white space then the spawned prefab is unparented (which may be what you want).

TLDR; want parent? Provide one or use the parentNameOrTag search string.

The keepWorldPosition parameter is passed to Transform.SetParent if a parent is provided or located. If true, the parent-relative position, scale and rotation are modified such that the object keeps the same world space position, rotation and scale as before (from the Unity documentation).

Collidables

A `Collidable` is a GameObject with a TpSpawnlink component on the main or root GO. The TpSpawnLink's IgnoreCollider flag must be false, the root GO must have a Collider or Collider2D component.

The GO is checked for this condition when its spawned from a prefab or when it is despawned. If this condition is true then the reference is added to or removed from an internal HashSet of these 'Collidable' Game Objects.

This is a way to keep track of spawned GameObjects which are not part of an Archived Tilemap in a TileFab. See how this is used in the Layout demo.

Events

- OnCollidableObjectSpawned: when a collidable GameObject is spawned.
- OnCollidableObjectDespawned: when a collidable GameObject is despawned.

Service Messages

The Spawner Service doesn't accept any Service Messages. However, it sends the following messages right after invoking the CollidableObject events.

- When Spawned: Command = "SPAWNER-ADD-PREFAB" and the ObjectPacket's UnityObject property has a reference to the spawned GameObject.
- When Despawned: Command = "SPAWNER-DEL-PREFAB" and the ObjectPacket's UnityObject property has a reference to the GameObject that will be despawned.
  - As usual: it makes no sense to cache this reference as it will become inactive.

Services that want to get these messages should add these two strings to their `AcceptableMessages` property return value.

Preloading

Preloading a pool can be done with Preload().

You can check to see if a Prefab is preloaded with IsPreloaded().

Reset

Resetting the Pooler (normally not needed) is done with (wait for it) ResetPools().

Notes

For maximum efficiency, it’s suggested that you add the TpSpawnLink component to the root GameObject of your Prefab. If it isn’t present, then the call to SpawnPrefab will add that component automatically so that the Prefab can be tracked.

TpSpawnLink provides optional timed auto-destroy. The OnTpSpawned and OnTpDespawned methods can be overridden in derived classes to provide customized activity when the Prefab is spawned or despawned. The DespawnMe method can be used to despawn a Prefab from some other entity.

If you do override anything that's `virtual` please be sure to properly call the base class methods from any overridden methods even if they appear to do nothing.

It’s a simple but functional system that can maintain the complete lifetime of a Prefab’s placement from a pool and a despawn with automatic return to the pool.

The System Info and Services Inspector windows show information about the pool status. Try out the Collision demo to see the pooling in action.

PoolHost

Normally the pooler does not attach the pooled and/or preloaded prefabs to a parent GameObject, which can make the hierarchy look messy.

If this bothers you, head over to the Project folder Plugins/TilePlus/Runtime/Assets and drag the Tpp_PoolHost prefab into your scene. This prefab has an attached component which sets

DontDestroyOnLoad so that the prefab persists between scene loads. The pooler looks for a GameObject with this specific name and will parent non-spawned and/or preloaded Prefabs to this GameObject.

GameObjects are set inactive when held in the Pooler after preloading or despawning but are set active when fetched from the pool.

Revision #12
Created 2025-06-22 20:11:01 UTC by Vonchor
Updated 2025-08-16 19:56:27 UTC by Vonchor