TSceneInitializer

You use one of the SetScene overloads of ChunkedSceneManager to change TScenes.

The very last thing that TpChunkedSceneManager.SetScene does prior to invoking the OnAfterTSceneChange callback is to evaluate all the Initializers for each TSceneSpec in the TScene and each Initializer referenced by the TpZoneLayout that's specified by name in the TSceneSpec.

Quite a mouthful. But what's a TSceneInitializer, anyway?

It's a Scriptable Object asset in your project. It must subclass TSceneInitializer. It has one method that you need to override: Exec(), and one serialized field: AugmentDefault.

TSceneInitializers are evaluated in a simple hierarchy: the TSceneInitializer referenced by the TpZoneLayout is the default and the TSceneInitializer referenced by the TSceneSpec (if any) is secondary: it can be used in addition to the default (augments) or instead of. The state of AugmentDefault is used to control what happens when there are two TSceneInitializers:

AugmentDefault on the TpZoneLayout's TSceneInitializer is ignored.

Possible cases:

- TSceneSpec has TSceneInitializer and so does TpZoneLayout
  - TSceneSpec TSceneInitializer has AugmentDefault = true
    - exec `TpZoneLayout TSceneInitializer`
    - exec `TSceneSpec TSceneInitializer` (augments whatever the ZoneLayout's TSceneInitializer does).
  - TSceneSpec TSceneInitializer has AugmentDefault = false
    - exec TSceneSpec TSceneInitializer
- only TpZoneLayout has TSceneInitializer
  - exec TpZoneLayout TSceneInitializer
- only TSceneSpec has a TSceneInitializer
  - exec TSceneSpec TSceneInitializer

In words, *if there's a TSceneInitializer attached to the TpZoneLayout it's the default and ChunkedSceneManager will always invoke its Exec method. If there's also a TSceneInitializer in a TSceneSpec it can be used in addition to the default (augment) or instead of the default, or, if there's no default at all then that one is used.*

What are TSceneInitializers used for?

Entirely up to you. Examine MainGridSceneInitializer in the Layout demo for an example.

When loading a level you often need to do further configuration based on what was loaded. For example, your level might have waypoints, and you want to have a list of where they are so that you can position your Player character at the last waypoint. You may have some special features that are in some levels and not others.

Often this devolves into having lots of tests specific to particular levels. Using initializers allows you to have post-load operations that are generic to all levels in the TpZoneLayout's referenced TSceneInitializer; sort of like a refactoring.

Then, for levels with specific initialization requirements, use another TSceneInitializer referenced by a particular TSceneSpec. The AugmentDefault setting on this secondary TSceneInitializer can be used to control whether it is used in addition to the default TSceneInitializer or instead of the default TSceneInitializer.

Let's examine the MainGridSceneInitializer with some added comments:

```
/// <inheritdoc
/>

/// <remarks>Here the passed-in object to the callback is the TpChunkedSceneManager
component</remarks>
public override bool Exec(TSceneList.TSceneSpec
tSceneSpec,
                          TpZoneLayout
zoneLayout,
                          TpChunkedSceneManager
sceneManager,
                          Func<TSceneList.TSceneSpec, TpZoneLayout, object?, object>?
callback)
{
```

```
    //get all the waypoints in this
Tscene
    var selector =
tSceneSpec.m_Selector;

    if (selector ==
null)

        return false; //should not
occur



    var zm =
zoneLayout.LayoutZoneManager;

    if (zm ==
null)

        return false; //should not
occur.



ChunkingDemoGameState.m_TemplateWaypoints.Clear();


ChunkingDemoGameState.m_TemplateWaypoints


.AddRange(selector.GetTilePlusTilesOfType<CdemoWaypointTile>(zoneLayout,

                                    WpFilter, 32));      \\FINDING ALL THE
WAYPOINTS


    //filter does nothing, use is illustrative
only.
    //the tile is an asset from a
TileBundle.
    //the string is the tilemap name embedded in the bundle's parent
```

```
TileFab.
    bool WpFilter(CdemoWaypointTile tile, string
s)

{

        return
true;


}




ChunkingDemoGameState.m_TemplateNpcSpawners.Clear();

    //get all spawners.  This is just as an example; the returned value isn't used in this
demo.
    ChunkingDemoGameState.m_TemplateNpcSpawners       FINDING ALL THE SPAWNERS

.AddRange(selector.GetTilePlusTilesOfType<NpcSpawnerTile>(zoneLayout));


     //FINDING ALL THE IMMORTALIZER TILES


    //get all Immortalizer tiles. This tile is a totally passive tile that describes an
area.
    //ALL layout zones in this area will be immortal ie won't be deleted
until
    //ALL zones become deleted on a change
scene.
    var immortalizerTiles =
selector.GetTilePlusTilesOfType<TpImmortalizer>(zoneLayout);


ChunkingDemoGameState.S_TemplateImmortalZones.Clear();
```

```
ChunkingDemoGameState.S_TemplateImmortalZonesLocatorPositions.Clear();

    //A hashset of all the locators. Hashset ensures no
duplicates
    foreach (var qr in
immortalizerTiles)



{

        var pos     =
qr.m_Position;

        var locator =
zm.GetLocatorForGridPosition(pos);


ChunkingDemoGameState.S_TemplateImmortalZones.Add(locator);

        ChunkingDemoGameState.S_TemplateImmortalZonesLocatorPositions.Add((Vector3Int)
locator.position);

}



    //now we have a hashset of the locator positions that are immortal. Faster, see loading
filter callback.
    //note that even if there are multiple immortalizer tiles in one zone the HashSet will
have only one entry.


    return
true;

}
```

You'll note the repeated use of selector.GetTilePlusTilesOfType. This is a very handy method that can be used to extract any particular Type of TilePlus tile. It's assumed that any interactive tile will be derived from TilePlusBase. There's no similar facility for normal Unity tiles.

Normally there aren’t that many TilePlus tiles in a TScene. Nevertheless, it’s recommended to only use this method during initialization as it has to scan through all of the TilePlus tiles each time that it’s called.

Revision #6
Created 2025-07-11 11:49:55 UTC by Vonchor
Updated 2025-08-13 16:58:34 UTC by Vonchor