TpLibInit and TpLib Memory allocation

The static library TpLib.cs has several Dictionaries that keep track of all TPT tiles.

The initial size of these dictionaries is set by constants in the TpLib.cs file. Similarly, pooled Dictionaries and Lists have a constant size when new pooled instances are created.

These constant size values are small. It is possible to change these allocations during App startup with the TpLib.Resize method. An instance of the TpLibMemAlloc class is passed to Resize.

One could change the constants themselves, however, updating the Plugin will naturally revert these values. Hence, Resize is a better choice.

There’s no reason to use this feature in an editor session. At runtime, use Resize immediately/soon after startup.

Internally, Resize releases all pooled items and resets MaxNumClonesPerUpdate and MaxNumDeferredCallbacksPerUpdate to their initial values.

The TpLibMemalloc values for pooled Dictionaries and Lists affect the size of new pooled instances of

- Dictionary<Vector3Int,TilePlusBase>
- List<TilePlusBase>

These are created and pooled frequently and optimizing this size can have a significant effect on performance.

TpLibInit

This is a scriptable object in a resources folder that you can use to set up memory allocations and certain optional features. It's in TilePlus/Runtime/Resources/Tp. TpLib startup code examines this asset and uses the following fields:

- Active: the asset is ignored if this is false.
- RefreshRequestsPerUpdate: Maximum number of Tile Refresh Requests per-internalUpdate

- MaxNumClonesPerUpdate: Maximum number of cloning operations per update.
  - Queued cloning requests come from asset-state TPT tiles added to a Tilemap.
  - Note that this value has no effect on the layout system.
- MaxNumDeferredCallbacksPerUpdate: Affects DeferredCallbacks and InvokeRepeatingUntil.
- TargetFrameRate: useful for diagnostics. Shown in System Info editor window.
  - If zero, adaptive features are disabled (not discussed herein)
- UsePlayerLoop: if true, TpLib modifies the PlayerLoop, if false, uses an Awaitables-based updater.
- ResizeMemory: if true, use the following info to resize memory.
  - MemAlloc: various fields for memory allocations. Read up before changing these.
- InhibitTilemapCallbacks. Normally false. Useful for debugging. Bad for production.
- Tweener Safe Mode: when checked, additional null checks are used. Advised to leave on.
  - If this is not checked and a tweened object becomes null there can be an exception.
  - Primary reason to uncheck this is for profiling.

PlayerLoop

Using the PlayerLoop timing is the best choice unless it inteferes with your project code somehow. When TpLib shuts down (app closes) or (Unity state change Play->Edit) it terminates the Awaitables-based updater OR restores the default Player loop.

If you are using PlayerLoop in your app and this doesn't work for you, subscribe to the `OnResetPlayerLoop` callback (it's not an Event, only one subscriber allowed.). The callback is invoked just before restoring the default Player Loop.

If the callback exists and returns false then the default Player Loop is restored. If the callback returns true then it's assumed that you handled this situation yourself and no further action is taken.

Revision #10
Created 2025-07-12 20:21:16 UTC by Vonchor
Updated 2025-09-20 13:07:16 UTC by Vonchor