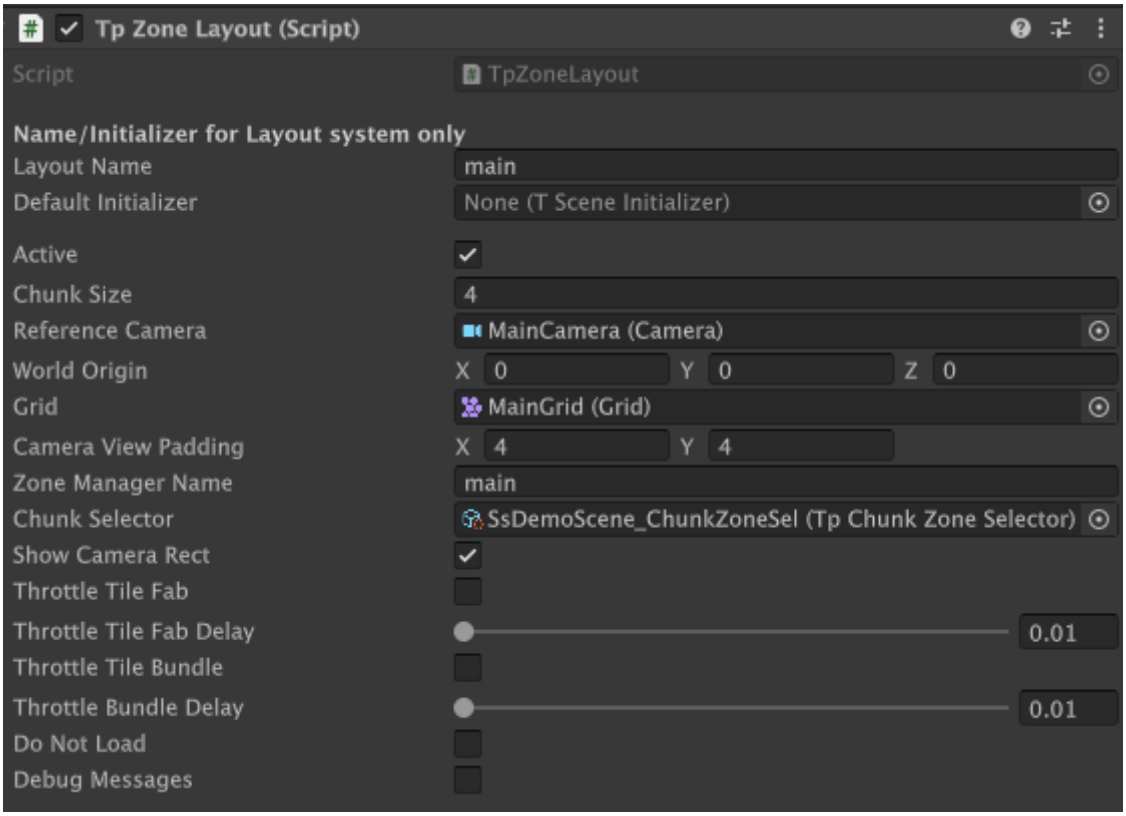
TpZoneLayout

At an even higher level, the TpZoneLayout MonoBehaviour component leverages a ZM to implement a basic camera Viewport chunking system. This component can be used as is, or as a base class for something more complex. Examine the “Layout” demo to see how to use it.

The following crude illustration shows an array of chunks on a Tilemap, some type of Player character, and a camera tethered to the Player, that is, a simple Top-Down view of a Tilemap.

ZoneLayout1.png

Example Zone Layout



ZoneLayout places TileFabs in the visible area of a camera viewport and deletes them from outside the viewport. The provided implementation only supports orthographic cameras. The location for each of the Chunks shown in the above illustration corresponds to a Locator maintained by a ZoneManager instance.

As the Player moves, so does the Camera. As the Camera moves, its Viewport moves, and overlaps different chunks after each movement, akin to a window sliding over the sGrid and the Tilemap. When a chunk moves out of view it should get deleted, and when there’s an empty chunk in the view then a Chunk should be placed there.

The TpZoneLayout component does most of the initialization work for you.

Let’s look at some of the ZoneLayout fields:

- LayoutName: A name for this ZoneLayout. Used with ChunkedSceneManager.
- DefaultInitializer: The default Initializer Scriptable Object asset. Used by ChunkedSceneManager.
- Active: Handy for debugging, if unchecked then this layout is ignored.
- Chunk Size: Size of chunks 4,6,8… This value is used to subdivide or 'Chunkify' the TileFabs and their Bundles.
- Reference Camera: the Camera you want to use for its viewport. In many situations this value will be the same for multiple layouts. But it doesn’t have to be.
- World Origin: the origin of the sGrid. Can be different on each layout.
- Grid: the parent Grid of the Tilemaps that this layout will be using. When using multiple layouts, it is important to ensure that this is the correct Grid for the Chunks that you’ll use.
- Camera View Padding: adds some extra space around the Camera viewport.
- Zone Manager Name: Must be unique for each Layout. If not, one of the duplicates will be ignored.
- Chunk Selector: A Scriptable Object asset that selects which Chunk to use at a particular Locator.
- Show Camera Rect: displays a marquee around the Viewport. Useful for debugging.
  - Not shown in the image above: Show Zones. When a layout pass occurs, draw boxes for the zones using chunk-sized squares showing how it is subdivided.
  - Note that when these boxes appear the Camera Rect display, if enabled, is temporarily not drawn.
- Throttle TileFab and Delay: add a delay between TileFab loads.
- Throttle TileBundle and Delay: add a delay between loads of a TileFab’s individual bundles.
- Do Not Load: No Chunks are ever loaded. You can watch the marquees move about without any distractions. Can get hypnotic.
- Debug Messages: exactly what it says...

TpZoneLayout uses a method in TpZoneManagerUtils to calculate a RectInt which describes the viewport.

Please note that if either 'throttle' checkbox is on, padding is automatically increased by (1,1) to avoid visual artifacts.

Since the viewport size is floating-point and a RectInt is not, the calculations always round up, so in general, the RectInt computed as the viewport size is bigger than the actual viewport. This is completely OK and is desirable as the effect helps to reduce visual artifacts.

Even with the rounding-up of the viewport size, you always need to add some padding so that there are no visual artifacts. This is very app dependent.

Then ZoneLayout uses the ZM to detect which chunks are within the viewport and which are not. Chunks outside the viewport are deleted from the Tilemap. Any empty Zones within the viewport are filled in with a Chunk provided by either a ChunkSelector asset or via a callback provided during initialization of the Layout (that’s not done in the Layout demo programs). The TileFabs can have filtering if you provide a filter callback.

If Prefabs Will Move

IMPORTANT: a TpTileBundle may contain prefabs which were parented to the archived Tilemap when the Bundle was created. Such prefabs are added to the Tilemap when the Bundle is loaded by TpZoneLayout and deleted when the Bundle is unloaded. However, these prefabs should not move. If they move into another Zone they won’t be deleted until that other Zone is deleted.

Hence, prefabs that will be moved should have a collider AND be spawned by using SpawningUtil (from a TpAnimatedSpawner or via code). Such prefabs will be tracked by SpawningUtil as ‘Collidables’ and callbacks are invoked when a Collidable prefab is added or deleted. Your application needs to keep track of these.

See the TopDown Layout demo to see how that works: The GameController's PreFilter and the GameState service are where the callbacks mentioned above are handled.

Special TileFab User Fields

Each TileFab has two user fields: a boolean and a string. These can be used for various things, dependent on your game structure. One use might be in the LoadingFilter callback passed to TpZoneLayout. UpdateTickAsync.

Selectors and ChunkSelectors

By now you’re wondering what this new bit of jargon is all about. A Selector is a bit of code that ZoneLayout invokes to obtain the Chunk to place at a certain location. At a low-level, a callback for this can be provided when initializing the ZoneLayout from code. A ChunkSelector is the same thing, except it is embedded inside a Scriptable Object so that it’s easier to use in the Unity Editor environment.

Since they’re essentially the same thing, we’ll just call them Selectors from now on.

ChunkSelectors use the IChunkSelector interface, which contains methods for initialization and Selection.

TpSingleFabChunkSelector couldn’t be simpler: it just returns the same Chunk every time. This is useful when you want to fill a solid background layer.

TpChunkZoneSelector is almost the same: it returns a single TileFab. However, internally it subdivides the archived tiles into areas of an arbitrary size, effectively precaching all the data required for quick loading to a Tilemap using that class’s bulk-move methods.

To dig into this more, see this for an extended discussion, including more information about a higher-level component called TpChunkedSceneManager.

`TpChunkedSceneManager` lets you load one or more Tilemap “Scenes” using the ZoneLayout/ZoneManager layout system.

You define Tile Scenes or TScenes using a specialized editor window.

The TpChunkedSceneManager MonoBehaviour component allows you to load TScenes by name, an index in a list of TScenes, or by the TScene’s GUID. Using the GUID is preferred for save files. ChunkedSceneManager also handles all the work involved in the creation and re-use of Zone Manager instances.

A flexible system of SceneInitializers lets the ChunkedSceneManager automatically invoke code that can extract information from the loaded tiles (e.g., waypoints, spawners) when the TScene is loaded.

Callbacks invoked at several stages of the loading process are used to

- Indicate that the current TScene is about to be unloaded.
- Indicate that a new TScene is about to be loaded.
- Indicate that the new TScene loading was completed so you can do any final initialization.

Revision #14
Created 2025-07-08 14:26:00 UTC by Vonchor
Updated 2025-09-22 21:07:41 UTC by Vonchor