

# Components

Monobehaviour components for TilePlus

- [TpBundleLoader, TpFabLoader](#)
- [TpSpawnLink](#)
- [Unity UI and New Input System](#)
- [Others](#)

# TpBundleLoader, TpFabLoader

These are components that can be added Tilemap's GameObject to load TileBundle or TileFab archives.

- TpTileBundle or TpTileFab: the primary asset which holds all the tile assets and their locations.
- LoadOnRun: automatically load when app runs. Uses LoadPrefabs and ClearMap settings.
- Offset: offset every item in the archive when loading. Normally 0,0,0.
- DelayTime: delay before refreshing the Tilemap in Play mode or in a built application.
- When the Asset field is populated, a Load button will appear.
- LoadPrefabs: if this is checked then runtime-loading or the Load button also loads prefabs in the asset.
- ClearMap: if this is checked then runtime-loading or the Load button clear the Tilemap before loading.
- ClearPrefabs: if checked runtime-loading or the Load button clears ALL GameObjects parented to the Tilemap.

If any Bundle was built with prefabs included, then clicking "Load" with the LoadPrefabs toggle checked instantiates the saved prefabs (or variant prefabs, if that's what you chose to save) if LoadPrefabs is true.

# TpSpawnLink

This component should be attached to any prefab that you use with the SpawningUtil pooling system. If you forget it, SpawningUtil will automatically add it for you, but that's a tiny bit slower.

If you add it to a prefab manually you can use some of its special features:

- Auto-Destroy after a timeout.
- IgnoreCollider: the spawning system adds any prefab instances which have a collider to a special internal list. If set TRUE, the spawning system won't add an instance of such a prefab to this list. Used primarily with the Chunking/Layout system.

This class can be extended, however please ensure to call base class methods if you override any methods.

# Unity UI and New Input System

Unity UI can invoke methods in monobehaviour targets using Unity events. That's not useful if you want, say, a button click to send a message to a tile.

If you're using the New Input System it can be handy to translate a mouse click or some other Action to locate and message a tile.

## TpGuidToAction

This component can be used as a target for a Unity UI control, it just sends a message to a tile that's specified by a GUID. You can see how to use this in the AnimatedTiles demo (Animation-UnityUI scene).


## TpInputActionToTile

This component converts New Input System actions into TPT tile locations and can send a message to a target tile if you click on it. An example can be found in the AnimatedTilesDemo (Animation-MouseClicksDirect) and in the Layout demo. It's also used for the ['UI system'](#)

This component is an easy to use front-end for a `TpActionToTile` scriptable object instance. Check out the source code for this class if you'd like to dig into what's going on. But generally the component is a better way to use this feature.

### [InputActionToTile.png](#)

This is a Ui-Elements custom inspector that expands and contracts fields as needed depending on how you set it up.

Click the  button for some quick help.

There's a lot to unpack:

The very top section is an area where you can specify one or more pairs of Camera and Tilemap. Cameras and Tilemaps go together since both of those components are needed in order to translate from screen coordinates to tilemap coordinates. You can have multiple pairs as needed, and a priority value can be used to select a tile target when two tiles overlap the same position.

The priority value is used when you have multiple tilemaps *unless* `Use Renderer Sort` is checked.

- If checked, priority is set with the Tilemaps' Sorting Layers and Order in Layer values.

If `No Messaging` is checked then evaluation stops prior to sending a message for Clicks (only) and only the `OnBeforeMessageSent` or `OnNoMessageSent` callbacks are used for this Tilemap.

If `Use Position Db` is checked then the PositionDb Service is used to locate tiles. This is useful when you have tiles with scaled sprites or if you're tweening position and/or scale.

In the image, Enable Hover is unchecked so the HoverAction name field isn't shown. Normally you don't care about Hover.

`Click On Mouse Down` a click = mouse down. If not, a click = mouse up (i.e., after button released).

`Handle Events`: If checked, handle **all** events in this component. This only works for cases where ALL of the event response is dictated by Event or Zone actions that can automatically be invoked by `TpEvents.ProcessEvents`.

`Emit Events`: If checked, the message sent to the tile specifies that it can post an event if that's appropriate.

The optional `Sound Controls` section lets you add audio feedback.

The `Ui/Messaging` section lets you add direction info to packets and show a prefab at the click position.

`Add Direction 4` and `Add Direction 8`: this is a little harder to explain.

These are used when you want to know *where* in the tile the mouse pointer actually was. This may sound strange but consider that a tile with an unscaled sprite essentially occupies one Tilemap unit. But the mouse click can be anywhere within that space.

The ActionToTile packet sent by this component includes an `offset` which is the Vector2 distance from the center of the tile to where the click actually was.

If an `add` box is checked then that Vector2 position is converted into an angle and then into a value from the Position4 (up/right/down/left) or Position8 (up, upright, right ....). The `Center Dead Zone` describes what part of the sprite's area is considered as `None` from the two Position enums.

So what does all that mean? If you enable this feature then a message recipient can determine if it was clicked in the center area or on one of the edges.

You can see this at work in the Oddities/Jumper demo, where clicking on the edges of a tile moves it in the direction of where it was clicked.

When using PositionDb this feature works correctly for scaled or position-shifted sprites, even if they no longer overlap the actual tile position.

---

Another way of thinking about this feature:

The packets have information about where within the tile the click actually occurred as well as 4 or 8-way direction info (meaning you don't usually have to evaluate the position within the tile).

The direction info is evaluated as a rotation clockwise from straight up. E.G. one can interpret `DirectionType4.Right` as 'move this tile to the right' or `DirectionType8.RightUp` as move diagonally up and to the right.

# Others

## Tilemap Parallax

Add it to a Tilemap, provide a follow target, and this component offsets the Tilemap's transform as the target moves. See the [Side-scroll Layout demo](#).

## SetDontDestroy

Add it to a GameObject and DontDestroyOnLoad will be set on Start().

## TpNoPaint

1. Add this component to a Tilemap's parent GameObject to prevent it being used for painting.
2. Add this component to the GameObject of a Palette prefab to inhibit reporting of # of different tiles in the palette.

## TpPrefabMarker

Used during the creation of TileBundles and TileFabs. You never need to use this yourself.

## AnimStateTilePingerBase

Yeah, annoying name. Use as a state machine behaviour to notify a TPT tile, e.g. a TpZoneAnimator.

- TpZoneAnimator pokes its instance ID into the animator as a parameter (called 'id').

Other uses of this class require a similar setup