The Big Deal

A Key Feature

None of the TpLib TMDB and query functions, and none of the Event, Messaging, Persistence, and most other TilePlus functionality would work without the special TilePlusBase class.

Throughout the documentation and codebase you'll see TilePlusBase tiles referred to as TPB. TPB and derived tiles are generically referred to as TPT tiles.

Here's a class diagram generated in Rider which shows the basic class structure for all the TilePlus tiles in the Runtime library:

TilePlusBase_ClassStructure.png

The lines show inheritance and the various dependencies.

TilePlusBase is divided into several partial classes just for organizational purposes:

- TilePlusBase
- TilePlusBaseData
- TilePlusBaseEnums
- TilePlusBaseStubs
- TilePlusBaseZoneEditor
- (Editor Folder) TilePlusBaseEditor.

Name the most powerful boss from any game you like: you would rather meet that boss in unarmed naked combat (IRL) then mess with this code. Seriously, read-only!

But unless you're deep into coding, you can ignore it. You can create a TilePlusBase tile asset and Paint it on a Tilemap but there's limited use for such an asset.

All the useful TPT tile types are derived from TilePlusBase, which, aside from supporting the use of cloned tiles properly, provides many properties and methods for the derived classes to use including basic animation support features.

Usually you'll only need to create derived classes from TpSlideShow, TpFlexAnimatedTile, or one of the two animated spawner variations.

Two important notes:

- Overidden methods generally should have their base classes invoked.
- StartUp should call the base class before doing anything else and return false if the base class returns false.

Basics

TPT tiles have three possible states maintained by the TilePlusBase instance.

- Asset: The tile is an asset in a Project Folder. Painting it changes state to Clone.
- Clone: The tile is present in a Scene. You can save it to the Project as an Asset.
- Locked: The tile is part of a TpTileBundle (Bundle) asset in the Project Folder.

If you Inspect a TPT asset in the Project window and open the “TilePlus Basic Settings” foldout, you’ll note that the State field is Asset.

When a TPT tile is added into a Palette it remains in the Asset state. You can see this by selecting it in the Palette Window – the Brush Inspector’s Tile Info/Name field displays `[Asset]`.

When the tile is painted on a Tilemap, the state changes from Asset to Clone. You can see this by picking the tile using the Palette Select tool and looking at the “TilePlus Data” section’s first line.

The only exception is when you use the Pick function of a Brush or Tile+Painter to copy and paste tiles. In that case, the copied tile is already a clone, and the pasted tile is the same clone, which is not what’s wanted. The system recognizes when this happens (in-editor only and Play mode if you copy/paste programmatically), makes a new clone of the tile and paints it in the pasting location. This is implemented by TpLib.CopyAndPasteTile.

You can use a Selection Inspector toolbar button to save a TPT tile from the Scene to the Project as a Normal, cloneable TPT tile asset. The selected tile isn’t affected. This is handy for prototyping: you can customize a TPT tile right in the Scene and save it either for backup or as a template for further use. A simple versioning scheme adds version numbers to the saved assets.

Locked TPT tiles are the same as ordinary tiles in the sense that any modifications to the Locked tile painted on the Tilemap affect the tile asset in the Project. They’re only seen as sub-assets of an asset created when the Tilemap bundling functions are used.

If a Locked tile is present in a tilemap, it converts into a Clone tile at runtime.

One of the key features of TilePlusBase tiles (and their subclasses) is that the Tile instance always knows what Tilemap it’s part of and always knows its position on the Tilemap. The position information isn’t static: the position is updated if you move a tile. To be clear, this is performed entirely within the tile and has nothing to do with how the tile was placed: Unity Tilemap Editor, Tile+Painter, or via code.

When a tile is placed or moved, it calls a method in TpLib to register itself in the various data structures. This also happens when you start your app or load a new scene.

Revision #9
Created 22 June 2025 19:07:40 by Vonchor
Updated 6 July 2025 14:18:59 by Vonchor