TileFabs and Bundles

Introduction

You probably work with Tilemaps that are part of a scene.

This is the easiest way to use TilePlus tiles: just paint and edit them to do what you want.

This long section of this book attempts to progress from the lowest-level concepts through those that are higher-level (such as Monobehaviours). What this means to you is that there's a lot of exposition coming up. Have patience!

Saved with a Scene

When you paint a TPT tile it’s saved with the scene and will be present in a build.

- The TPT tile in the scene is not affected by changes to the asset in the Project folder.
- The asset in the Project folder is not affected by changes to the tile in the scene.

If you paint over the TPT tile, it’s deleted from the scene and will no longer be saved. Essentially, the TPT tile becomes like any other Object in the scene.

Once you edit the painted TPT tile in-scene you can save it as a new tile asset, drag that into the palette or Painter’s Favorites list, and paint it. This is especially useful when prototyping or for creating backups.

This is the easiest way to make the most of this new type of tile. Remaining in the scene, you can have references to Scene Objects like GameObjects, Components, etc. in your tile scripts. Once a tile is painted, you can select it with the Palette, edit it in the Tile+Brush Selection Inspector or Tile+Painter, and drag references such as GameObjects into the painted TPT tile fields. Think of it as a Tile promoted to a GameObject, just without a Transform component. That’s close enough for rock and roll.

Scene References

When a TPT tile is in the Scene, it can have references to other Objects in the Scene. For example, you can have a reference to a GameObject and access its components for scripting, right within the TPT tile’s code if you like.

BUT: maybe you want to make prefabs out of Tilemaps and load them while your game is running.

If so, you may find this system annoying. There is a limited facility to save Tilemaps as Prefabs as explained in the next section. However, it’s way more efficient to use the TileFab system. Tilefabs

allow preserving one or more Tilemaps and any child prefabs into a position-independent archive format.

TileFabs are the key technology component underlying the TilePlus layout system, which streams chunks of tiles in and out of the scene as the Camera moves.

Why Not Make Prefabs?

It's a bad solution

When you create a prefab by dragging a Grid with child Tilemaps to the Project folder, all references to Objects in the Scene are lost just like any other Prefab that you might create. This includes all TPT tiles. If you were to open the Prefab, the locations where TPT tiles had been placed would be replaced with pink or other oddly colored tiles.

TilePlus Won't Let You

If you mouse-drag a Tilemap or any GameObject that has Tilemaps as child GameObjects into a project folder to create a prefab, and there are any TPT tiles, the system will warn you in the console and will unlink the items you dragged from when you created this prefab so that the scene Tilemaps won’t be corrupted. You may as well delete the prefab that you created as it isn’t useful.

You Probably Don’t Want Tilemap Prefabs

They’re not really that useful except in limited circumstances. Each time you drag in a Tilemap prefab, it instantiates an entirely new Grid with Tilemap children. This is true for any Tilemap prefab, even one created normally by dragging a Grid GameObject from a scene to a project folder.

What’s more useful is being able to load tiles in groups to existing Tilemaps. You may already do this with Tilemap block move methods and so on.

TilePlus’ TileFabs and Bundles handle all this for you and are easy to create as you'll see.

TileFabs and Bundle assets are not Prefabs!

Please note that the TileFab and Bundle assets are NOT prefabs: you can’t drag either of these into a scene.

However, a TileFab archives the original source Tilemap names and/or tags.

The TilePlus library functions for placing TileFabs in a scene expects to find these same names and/or tags to place the multiple archived Tilemaps correctly. That is, which Bundle assets referenced by a TileFab should be ‘painted’ on to which Tilemap. It can’t guess.

Bundling Workflow

To make a compatible prefab:

- Use the Grid Selection mode of TilePlus Painter

OR

- Select a Grid with child Tilemaps or a single Tilemap,
- Make a Grid Selection using the Tile Palette or Tile+Painter.

Then use the Hierarchy window's TilePlus Bundler context-menu command or the main menu's Tools/TilePlus/Prefabs/Bundle Tilemaps command.

If a Grid or a single Tilemap is selected, then all the tiles and child Prefabs of the Tilemaps are bundled.

If a Grid Selection exists, you'll be prompted to confirm that you want to use the Grid Selection to limit which tiles and Prefabs are bundled. If so, ALL tiles and Prefabs in *all* of the Grid's child Tilemap(s) within the selection are archived.

This command archives Tilemap contents into one or more TpTileBundle assets (Bundles, from now on) and optionally creates a special type of prefab. But at this point its use is restricted to simple situations:

- Archive a single selected Tilemap into a Bundle asset along with a TileFab asset.
- Archive a single selected Grid and its child Tilemaps into multiple Bundle assets along with a TileFab asset, with optional prefab creation.

To be clear: Compatible prefabs can only comprise a Grid with child Tilemaps and any Prefabs parented to the Tilemaps. If you try to use a multiple selection, or any Grid or Tilemap are in or part of a prefab, then the process will complain and quit.

The bundler generally follows this process:

- Is the Grid or Tilemap in a prefab or are you editing in a preview scene? Start again.
- If the selection was not a grid, you're asked if you want to continue. No: Start again
  - If the selection isn't a grid, then you can't make a prefab, just archives.
- For each tilemap:
  - Is the Tilemap part of a Prefab? Start again.
- Select destination folder. Advice: use a different folder each time!
  - Is the folder the Assets folder? Start again.
- If the selection was a grid, you're asked if you want to make a Prefab of the Grid and all child Tilemaps (a Tilemap prefab).

    - If making a Prefab, you’re asked if you want to bundle any Prefabs which are children of the Tilemap GameObjects.
- You can provide a 'base' name for the generated assets.
- If the selection was a single Tilemap, you’re asked if you want to add any Prefabs which are children of the Tilemap GameObject, and if those should be saved as new Prefabs or Variant Prefabs.

It’s simpler than that sounds! The choices are presented to you via several dialog boxes and file pickers.

With this information, the bundler creates the TpTileBundle assets: one for each Tilemap, then adds all the TPT tiles to the asset. For Unity tiles, it preserves the information in the Tilemap for each tile (color, transform matrix, and flags) as well as a reference to the original Unity tile asset.

If you elect to archive prefabs, please note that references to the prefab assets are archived; new copies of the prefabs aren’t created. This means that if you delete one of these prefabs then the reference is null, and the prefab won’t be available when the TpTileBundle asset is used.

Note that GameObjects that are instantiated because the Tiles’ GameObject field caused the Tile to instance a GameObject into the scene are not archived. When the TpTileBundle asset is painted (programmatically or via the Tile+Painter) the Tiles re-create the prefab in the scene for you.

A TileFab asset is also created in the same folder. This asset has references to all the new TpTileBundle assets and is used with the TpAnimZoneLoader tile or for loading using methods in the TileFabLib library.

If you’re making a Tilemap prefab, it’s placed in the folder that you selected earlier. Don’t delete the TileFabs and Bundles created during the bundling process: the Prefab requires them.

The created prefab can be used just like any other prefab with one exception: it’s “Locked,” and the TilePlus tile instances are also “Locked.”

This locked status is a hint to Tile+Painter and the Tile+Brush to use particular care when it encounters TilePlus tiles in a Prefab.

Essentially, you can’t edit this sort of prefab in a Prefab editing context or “Stage” to avoid any chance of corruption, and you can’t edit the TPT tiles or the prefab itself. Regardless of which brush you use, painting, erasing, or any other operation will either revert or just not occur. You’ll also find that you can’t open one of these prefabs in a prefab editing context. There’s a discussion about why this is necessary in the online Programmer’s Guide. Tilemaps with locked tiles display a closed-lock symbol in the hierarchy window.

However, you still have the original Grid and/or Tilemaps; unlike normal Prefab creation via Drag and Drop, the source isn’t linked to the created prefab. If you ever need to recreate the original for editing just drag the Prefab into a scene and use the Unity menu command “Prefab/Unpack Completely.”

A small component called TpPrefabMarker is added to the Prefab, specifically, to the same GameObject as the Grid. Please don't remove it UNLESS you unpack the Prefab. If you do unpack, REMOVE the component. If you don't then it will overwrite all the tiles with tiles from the TileFab whenever your game starts running. At best, a time waster but at worst, you'll get profoundly confused. Here's why:

- When a Prefab is dragged into a scene or instantiated at runtime, PrefabMarker loads the tiles from the TileFab on to the appropriate Tilemaps. This component can be seen at the top-level of the Prefab in the Project folder. You'll see a reference to the TileFab. While you can change this to a different TileFab, be aware that the unpopulated Tilemaps themselves are in the Prefab and would need to be compatible with whatever TileFab you change to. The BundleLoadFlags field should usually be set to Normal. Use of other settings should be experimental only.

It's important to remember that the Locked Tilemap shouldn't be edited, moved, picked, flood-filled, etc. The system will actively try to thwart you from doing so, but you can finagle your way around it if you try hard enough. If you do manage to edit a Locked Tilemap somehow and save the prefab overrides, then the prefab can become corrupted and unusable, even if no TPT tiles were modified.

Details

As mentioned before, TpTileBundle (Bundle) assets are used to archive all or a section of a Tilemap. TpTileFab (TileFab) assets combine references to several Bundles, creating an archive of one or more Tilemaps; a multiple-Tilemap prefab of a sort.

In Bundles, cloned tiles in the scene are changed to Locked tiles and are stored as sub-objects of the Bundle asset. In other words, the cloned tiles are changed to Project folder assets and stored as part of the Bundle asset.

When normal Unity tiles are archived, just their transform, color, and flags settings are preserved, along with a reference to the tile asset in the project folder.

Prefabs which are children of the selected Tilemaps are archived by reference only: no asset copying occurs.

Note that Prefabs can be bundled from a Scene hierarchy or from a Project folder. This implies differences in how the transform information is archived:

- When bundling Prefabs from a scene, the stored rotation and scale are the rotation and localScale of the root GameObject of the Prefab in the scene hierarchy.
- When bundling Prefabs from a project folder, the stored rotation and scale are the rotation and lossyScale of the root GameObject of the Prefab in the Project folder.

Just to be clear about naming: Tilemap Prefab (with upper-case P) refers to a created prefab encompassing one grid with one or more child Tilemaps.

It’s encouraged to use a different folder each time you create Bundles, TileFabs, or Tilemap Prefabs, although this isn’t enforced.

Associated Components

You can use the TpBundleLoader component to load a single Bundle to a Tilemap. Place the component on any compatible (same layout, etc.) Tilemap’s GameObject and drag in the Bundle asset reference. Switch to Play mode and loading will happen automatically. Or you can click the Load button if you wish to test or perhaps restore a Tilemap.

If you maintain references to several Archive assets in your Scene (to ensure availability in a build) then you can change the asset reference in TilePlusLoader and call the Load method of the component.

Similarly, you can use the TpFabLoader component to load a TileFab.

Here are two other ways to use TileFabs, plus there's a lot more about them as you read on in this section.

- Paint a TileFab (or a single Bundle) using Tile+Painter. This is discussed in the Painter documentation.
- Load all or some of the Bundles in a TileFab on to different Tilemaps in your scene at runtime.

The second use listed above is particularly useful. Coders can use methods in TileFabLib to load one or more Tilemap sections dynamically from Bundles and TileFabs. For example, the TpAnimZoneLoader tile can be used to specify TileFabs to load when an entity passes into or out of a trigger zone. The TileFabLib static library has comprehensive methods to load TileFabs and Bundles, and enables higher-level elements such as ZoneManagers and ZoneLayout, the key elements of the Layout (Chunking) system.

The Bundling Process

When you use the Bundle Tilemaps menu command, the bundling process asks you some questions, as outlined in the User Guide. Using this information, the bundler creates the TpTileBundle (Archive) assets: one for each Tilemap. Then creates copies of all the TPT tiles and adds them to the asset, saves any asset references to Unity tiles, and saves all the information required to rebuild a Tilemap, including position, transform, color, and flags.

Since the values for transform, color, and flags are often the same for large numbers of tiles (especially so for Unity tiles), these are stored in indexed look-up tables.

After all the Bundle assets are created, a TileFab is created in the same folder. References to the Bundle assets are stored in this new asset.

If you're making a Tilemap Prefab of a Grid and its child Tilemaps, the bundler creates a new prefab with the Grid and the child Tilemaps. These child Tilemaps are empty. The parent Grid of the Tilemaps has the TpPrefabMarker component added with a reference to the created TileFab. The TpPrefabMarker component loads up the empty Tilemaps when the Prefab is dragged into the Scene or otherwise loaded.

If there are any prefabs as children of the Grid or Tilemap GameObjects then the Project folder references to these prefabs are added to the TpTileBundle.

To reiterate:

- When bundling Prefabs from a scene, the stored rotation and scale are the rotation and localScale of the root GameObject of the Prefab in the scene hierarchy.
- When bundling Prefabs from a project folder, the stored rotation and scale are the rotation and lossyScale of the root GameObject of the Prefab in the Project folder.

The completed Tilemap Prefab is placed in the folder that you selected earlier.

If you're archiving a single Tilemap into a Bundle, it's mostly the same process except that a Tilemap Prefab isn't created.

Tilemap Prefabs can be used just like any other prefab with one exception: it's "Locked," and you can't edit the Tilemaps. In Editor sessions, the system will try to stop you opening or making any changes to a Tilemap Prefab.

Why? A Tilemap Prefab contains copies of the Tilemaps, with no tiles. When the Prefab is placed in the Scene, TpPrefabMarker loads the tiles and prefabs from the TileFabs and Bundles. If you modify the Prefab in the Unity Editor by painting a TilePlus tile, then save prefab overrides, the new tile (being a scene asset) can't be saved in the Tilemap Prefab for reasons described earlier.

One could use the Allow Prefab Editing configuration option and paint normal Unity tiles or add GameObjects. That will preserve these changes in the Tilemap Prefab.

However, it's better to unpack the prefab, modify it, and generate a new Tilemap Prefab for maximum compatibility. You still have the original Grid and/or Tilemaps; unlike normal Prefab creation via Drag and Drop, the source isn't linked to the created prefab. If you ever need to recreate the original for editing, drag the Prefab into a scene, use the Unity menu command "Prefab/Unpack Completely."

Why does the Bundle Tilemaps command insist on a parent Grid to make a Prefab? If you make a prefab out of a Tilemap without a parent Grid, then instantiating it won't work properly unless you parent the instance to a Grid (this has nothing to do with TPT tiles). It will look fine in the mock scene created when you edit a prefab, because Unity adds the parent Grid for you in the editing Stage scene.

The created Archive asset contains all the converted, Locked tile assets. A name for the asset is created from the base name that you provide in the dialog box and/or from the scene and Tilemap names.

This is just for convenience so that you can have an idea of where the asset was created from.

If you inspect the asset, you'll notice that it has a few fields:

- Time Stamp: The creation time, in UTC time.
- Scene Path: The Scene path with dot notation.
- Original Scene: The name of the scene that the asset was created from.
- A flag that informs whether this Bundle was created from a Grid Selection.
- A flag that controls whether Tile+Painter includes this asset in its painting sources list.
- A User flag (a boolean) and a user string. Optional use.
- An Icon reference. Used to display an Icon for the asset in Tile+Painter.
- A GUID. This is used in the Layout/Chunking system.
- Various asset lists.

The Time Stamp, Scene Path, and Original Scene aren’t used in this distribution (but are used in the project that TilePlus was developed for).

If you know that you’ve edited a locked Tilemap, then to ensure that there are no issues, use the Tools/TilePlus/Prefabs/Unlocked Tiles test after selecting a single Tilemap. If there are clone tiles in a Prefab, you’ll have issues.

What’s the difference?

- A Tilemap Prefab instantiates Tilemaps and loads all the tiles from Tilefabs.
- A TileFab’s tiles are loaded on to an existing Tilemap and never creates new Tilemaps.

It’s a big difference! When a Tilemap Prefab is instantiated, a new set of Tilemaps is created and parented to a Grid. Therefore, if you instantiate 10 of these you have 10 independent sets of Grids and Tilemaps.

TileFabs require a set of compatible Tilemaps to exist in advance. So, if you have a scene with some Tilemaps and load a TileFab, the tiles load onto the existing Tilemaps. If you load the same TileFab 10 times the same tile would be written 10 times to the same locations.

The power of TileFabs comes when you consider that they can be painted anywhere on the Tilemaps. If you offset the placement of each of the 10 TileFab loads, they can be used to fill-in an area of a Tilemap. That can’t be done with Prefabs. A bonus is that Bundles and TileFabs can be painted with Tile+Painter.

Tilemap Prefabs can be useful as a quick way to load part of a scene. But you can also do that with TileFabs, and they’re way more flexible as they can be edited whilst being loaded.

Notes

Other Uses for Bundled Tilemaps

When using Tile+Painter, Bundle assets and TileFabs appear in the Painting Source (center) column and behave as if they were a single tile: you can paint them onto Tilemaps. This means that you can create chunks of tiles and paint them as if they were a single tile. For Bundles, you can paint the entire Bundle or view a list of all the tiles in the Bundle and paint them individually. These features are discussed in the Tile+Painter Guide.

If a TileFab is created from a Grid Selection then you can almost think of it as a rectangular piece of layer cake, with the Tilemaps being the layers. As just mentioned, you can then paint multiple pre-filled layers with one mouse click. Or you can use the supplied basic chunking system to move chunks of tiles in and out of a set of Tilemaps as they move in and out of the Camera view. Chunking is only supported on Orthographic cameras and only tested on the default Tilemap layout, that is, a top-down view.

For simpler uses, you can add the TpBundleLoader component to any Tilemap's GameObject. This component loads a single Bundle. Place the component on any compatible Tilemap (same layout, etc.) and drag in the Bundle asset reference. Switch to Play mode and loading will happen automatically if the component's LoadOnRun toggle is checked. Or you can click the Load button to make a quick test.

The Load button is hidden if the Tilemap is part of a Prefab.

Similarly, you can add the TpFabLoader component to any Grid's GameObject. It works in the same fashion as TpBundleLoader. However, it can't work correctly if the Grid's child Tilemaps do not match the names and/or tags of the Tilemaps embedded in the TileFab asset. That's up to you.

Finally, you can use a TpBundleTile. This simple tile takes a Bundle reference as a parameter. You can copy it to a Unity Palette or to Painter's Favorites list and paint it on a Tilemap. When you do, the tile loads the bundle and deletes itself. This tile is discussed more fully in the Tile+Painter User Guide.

Bundles and TileFab assets can be used in a running game to dynamically load tiles into Tilemaps using the TpBundleLoader or TpFabLoader components, a TpBundleTile, TpAnimZoneLoader tile, the TpZoneManager, or at the lowest level, the TileFabLib and TpZoneManager libraries.

One other note: When you make a prefab or archive, all Scene references are lost as usual. TPT tiles can have Scene references but if your tile doesn't have any then this doesn't matter. To be clear, any Scene reference within the same Prefab should work properly.