Rotorz Tile System

Editor Extension for Unity

Migration Guide

Migrate from version 1.x to version 2.0.0

ROTORZ LIMITED

2013-06-21

Table of Contents

Introduction

Welcome to this migration guide which explains how to upgrade projects that use earlier versions of the Rotorz Tile System extension to version 2.0.0.

Important

Always backup your files before updating to newer versions of Rotorz Tile System. You should also backup your files whenever updating or unpacking other assets.

Unity 3.x users can upgrade from any prior version of Rotorz Tile System. Unity 4.x users must upgrade from version 1.1.3 of Rotorz Tile System because older versions are not compatible with Unity 4.x.

Rotorz Tile System 2.x was designed to be unpacked alongside earlier versions to assist with the process of upgrading projects. This allows both versions of the extension to coexist during the upgrade process.

An upgrade wizard is provided to upgrade brushes and tile systems from any prior version of Rotorz Tile System. In summary the following steps are necessary to upgrade from an older version of the extension:

1. Backup your files.
2. Download and import version 2.0.0 of Rotorz Tile System.
3. Select menu **RTS** | **Editor Windows** | **Upgrader**.
4. Upgrade brush assets into new asset representation.
5. Upgrade existing tile systems.
6. Carefully remove older version of extension.

Redesigned User Interfaces

The various user interfaces have been overhauled with lots of new features, plus Rotorz Tile System has never looked better with the Unity Pro dark skin!

One of the most notable changes for existing users is that the main tool window has been split into three separate palette windows. This approach allows users to arrange these palettes to suit their workflow.

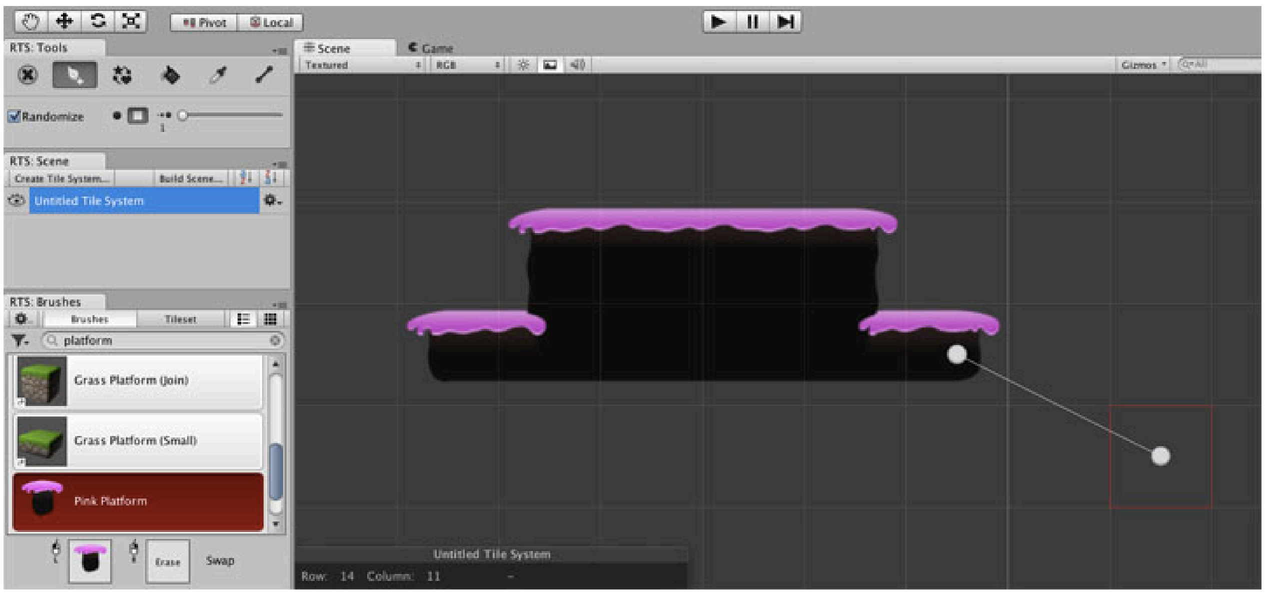


Figure 1. New design for palette window interfaces

Tip

If you have become accustomed to the original palette window have no fear, enable "Classic Mode" via the preferences window to combine the **Tool** and **Brush** palettes into a single editor window.

The designer window has also been revamped and now includes the ability to better manage tilesets and tileset brushes (formerly called atlas brushes). Tilesets can be created using the **Create Brush** window.

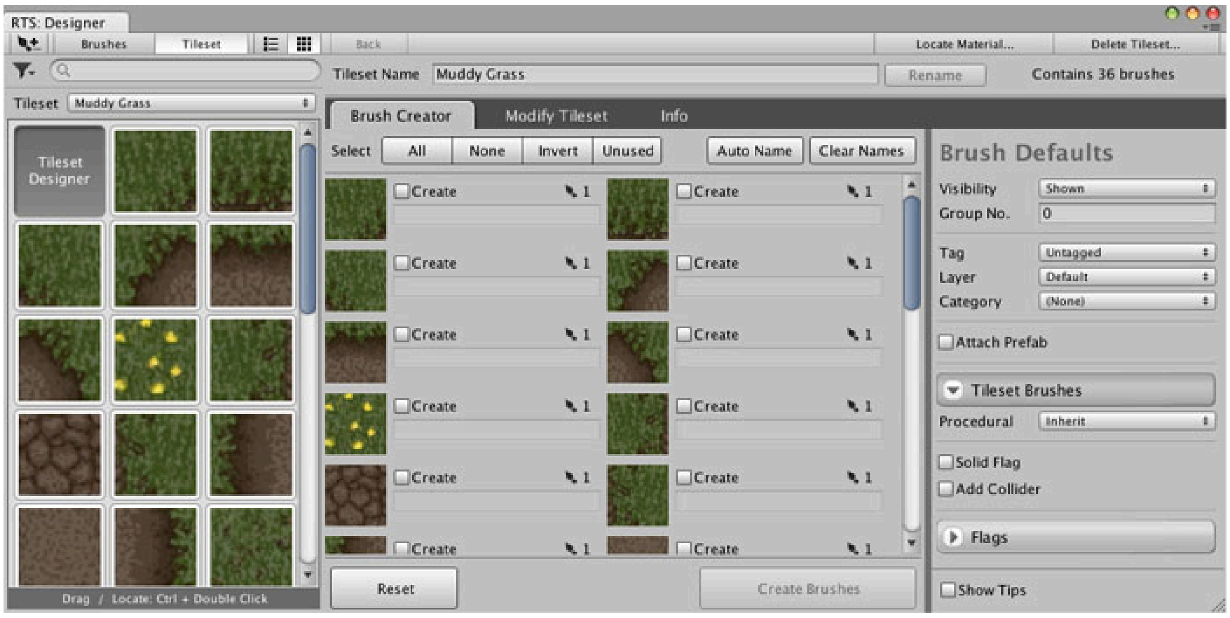


Figure 2. Tileset designer interface

New Brush Architecture

The method of storing brush assets has been changed entirely allowing for new features and better performance.

Brushes are now represented by scriptable object assets instead of game object prefabs. The included upgrade wizard can automatically convert existing brushes from their old representation into their new representation.

Tip

If you have previously associated brush prefabs with custom scripts, you will need to update your scripts and manually associate the new brush assets.

Basic Brushes

Basic brushes were created manually and thus this section does not apply to users who have only created brushes using the brush designer.

The concept of basic brushes has been removed since the same functionality can be achieved more easily using oriented brushes instead. Any existing brushes that use the `BasicTileBrush` script will be automatically converted into oriented brushes upon upgrade.

The prefab part of basic brush assets will be stored in the following path when applicable:

```
Assets/TileBrushes/Migrated
```

Upgrade Options

This section explains the options that are provided when upgrading brushes that were created using an earlier version of Rotorz Tile System.

Use procedural tilesets / atlas brushes

This option is only applicable to users who have created atlas brushes.

Previously atlas brushes were defined as prefabs with pre-generated mesh assets. Tiles were painted by instantiating the applicable prefabs. This meant that each painted tile was an individual game object which could be positioned, rotated and scaled independently of one another.

You now have the choice between creating non-procedural tilesets and procedural tilesets. Non-procedural tilesets are essentially the same as with previous releases of Rotorz Tile System.

Tiles that are painted using procedural tileset brushes are represented using procedurally generated meshes on a per chunk basis. This approach can offer signficiantly better performance when painting and erasing tiles whilst avoiding the need for pre-generated mesh assets. Additional game objects and colliders can still be attached to painted tiles if needed.

By selecting this option, previously defined atlas brushes will be upgraded into procedural tileset brushes. Do not select this option if you require the ability to position, rotate or scale individually painted tiles.

Tip

You can switch between procedural and non-procedural mode using the designer window at a later stage if needed. This option can also be overridden on a per brush basis if needed.

Use newer smooth platform brushes

Rotorz Tile System includes three "Smooth Platform" master brushes which can be easily reskinned with custom artwork to create custom platform tiles. "Grass" and "Cave" brushes are included to demonstrate the "Smooth Platform" master brushes.

These brushes are still included in version 2.0.0 of Rotorz Tile System. Select this option to use the freshly packaged version of these brushes, or deselect to upgrade the previous version of these brushes.

You may want to deselect this option if you have made alterations to the original brushes.

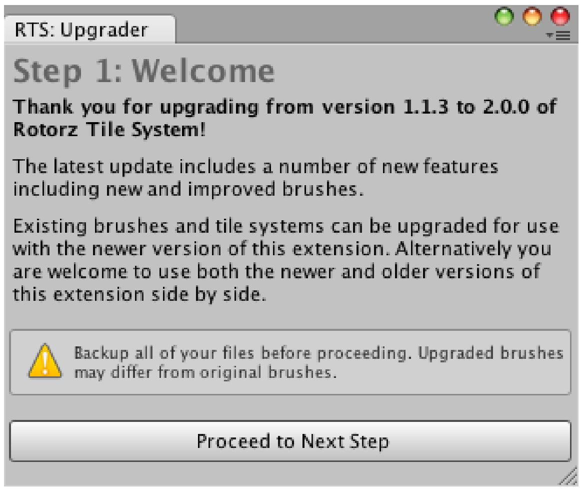
Note

If you deselect this option you may find that the "Grass" and "Cave" brushes appear multiple times in the brushes palette window. This occurs when the new version of the smooth platform brushes are unpackaged when importing the new version of Rotorz Tile System.

Upgrade Existing Brushes

Before you begin

Ensure that **Upgrader** window is shown by selecting **RTS** | **Editor Windows** | **Updater**. You should see something like the following if your project contains an older version of Rotorz Tile System.


RTS: Upgrader

Step 1: Welcome

Thank you for upgrading from version 1.1.3 to 2.0.0 of Rotorz Tile System!

The latest update includes a number of new features including new and improved brushes.

Existing brushes and tile systems can be upgraded for use with the newer version of this extension. Alternatively you are welcome to use both the newer and older versions of this extension side by side.

Backup all of your files before proceeding. Upgraded brushes may differ from original brushes.

Proceed to Next Step

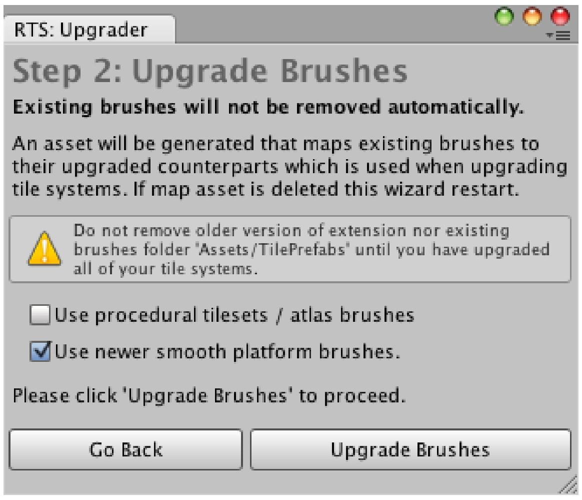
Figure 3. First step of upgrader window: Welcome

Procedure

1. **Very Important** - Backup all of your files before proceeding.

2. Read through the welcome message and then select **Proceed to Next Step**.

   You should then see the following interface:



RTS: Upgrader

Step 2: Upgrade Brushes

Existing brushes will not be removed automatically.

An asset will be generated that maps existing brushes to their upgraded counterparts which is used when upgrading tile systems. If map asset is deleted this wizard restart.

Do not remove older version of extension nor existing brushes folder 'Assets/TilePrefabs' until you have upgraded all of your tile systems.

- [ ] Use procedural tilesets / atlas brushes
- [x] Use newer smooth platform brushes.

Please click 'Upgrade Brushes' to proceed.

Figure 4. Second step of upgrader window: Upgrade Brushes

3. Read through prompt and adjust upgrade options as desired.

   See Upgrade Options on page 3 for further information regarding these options.

4. Select **Upgrade Brushes** to upgrade existing brushes.

   Please be patient as it may take a little while to upgrade your existing brushes.

   Brushes that were created using older versions of Rotorz Tile System are located within the folder `Assets/TilePrefabs` and are not automatically removed from your project because they are still required in order to upgrade existing tile systems.

   Tip
   Brushes that are created using Rotorz Tile System version 2.0.0 are placed in `Assets/TileBrushes`.

Results

Your brushes should now be available for use with version 2.0.0 of Rotorz Tile System. You can now proceed to upgrade your existing tile systems; see Upgrade Existing Tile Systems on page 8.

Some developers may be interested to learn more about the brush mapping asset which is generated when brushes are upgraded; see Brush Mapping Asset (Advanced) on page 5.

Brush Mapping Asset (Advanced)

Experienced developers can implement custom editor scripts to automate the remapping of brush assets in custom scripts.

A special asset `Assets/TileBrushes/BrushMappingsV1toV2.asset` is automatically generated once brushes have been upgraded which maps former brush prefabs to their new upgraded counterparts.

Example: Custom script with brush variable

You may benefit from implementing a custom editor script to remap brush references if you have a custom script that is attached to lots of objects in your project. This process can be done manually if you only have a handful of changes to make.

Here is an example of a custom script which makes reference to a brush:

```
using UnityEngine;
using Rotorz.TileSystem;

class SomeCustomBehaviour : MonoBehaviour {

    public TileBrush brush;

    public bool IsSolidByDefault {
        get { return brush != null && brush.solidFlag; }
    }

}
```

Currently the **`brush`** variable is only able to reference older brush instances. We can change the variable type from **`TileBrush`** to **`Object`** so that it can make reference to either an older or newer brush asset.

Once the variable type is changed you are likely to encounter a number of errors in your script. For now temporarily surround any broken code with comment markers. These errors can be resolved more easily once the variables of your existing objects have been updated.

CAUTION

Do not turn any class variables into comments to avoid data loss.

```
using UnityEngine;

class SomeCustomBehaviour : MonoBehaviour {

    // Still refers to old brush prefab!
    public Object brush;

/*
    public bool IsSolidByDefault {
        get { return brush != null && brush.solidFlag; }
    }
*/
}
```

Here is a custom editor script which will remap the brush variable for each selected object that has your custom script attached:

```
using UnityEngine;
using UnityEditor;

class CustomUpgradeScript {

    [MenuItem("Custom Tools/Upgrade Selected Objects")]
    public static void UpgradeSelectedObjects() {
        Object[] objs = Selection.GetFiltered(
            typeof(SomeCustomBehaviour),
            SelectionMode.ExcludePrefab | SelectionMode.Editable
        );

        RtsUpgradedBrushMap map = RtsUpgradedBrushMap.BrushMappings;
        if (map == null)
            return;

        foreach (SomeCustomBehaviour custom in objs) {
            // Lookup new brush asset from former brush prefab
            custom.brush = map.Lookup(custom.brush);
        }
    }

}
```

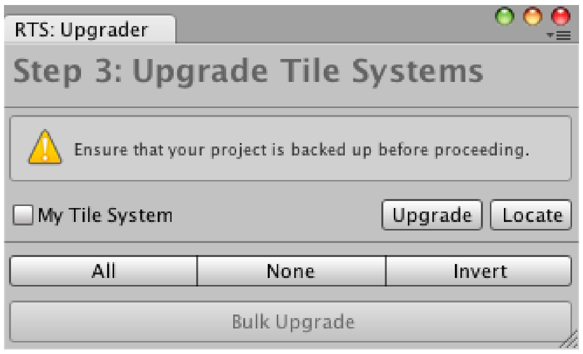
You can now apply the update to each of your objects. Once you have finished updating your objects you can remove your custom upgrade script and then update your custom script:

```
using UnityEngine;
using Rotorz.Tile;   // Let's use the new namespace

class SomeCustomBehaviour : MonoBehaviour {

    // Now refers to the new brush asset!
    public Brush brush;

    public bool IsSolidByDefault {
        get { return brush != null && brush.SolidFlag; }
    }

}
```

Upgrade Existing Tile Systems

Before you begin

Ensure that **Upgrader** window is shown by selecting **RTS** | **Editor Windows** | **Updater**. You should see something like the following if you have already upgraded your brushes and your project contains an older version of Rotorz Tile System.



Ensure that your project is backed up before proceeding.

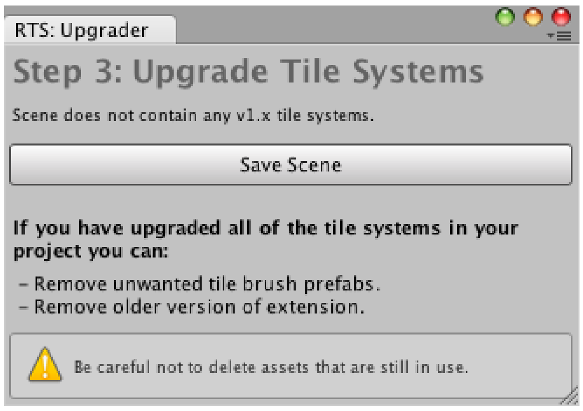
Figure 5. Third step of upgrader window: Upgrade Tile Systems

The upgrader window lists tile systems in the current scene that were created using an older version of Rotorz Tile System.

Procedure

1. Upgrade tile systems as desired.
    - Select **Upgrade** button to right of tile system that you would like to upgrade.
    - Tick boxes at left of each tile system that you would like to upgrade and select **Bulk Upgrade**.

   You should then see something like the following:



Scene does not contain any v1.x tile systems.

If you have upgraded all of the tile systems in your project you can:
- Remove unwanted tile brush prefabs.
- Remove older version of extension.

Be careful not to delete assets that are still in use.

Figure 6. Final step of upgrader window: Upgrade Tile Systems

2. Save the current scene.

Update Scripts

We hope that the process of updating your custom scripts is reasonably straightfoward. A lot of effort has been taken to provide detailed API documentation.

Whilst we are unable to update your custom scripts for you, please contact us if you experience difficulties and we will do our best to answer your questions!

Naming Changes

A number of changes have been made to the API including new namespaces, naming changes and due to popular request a more standardised naming scheme for class properties.

The biggest change to the new API is that the namespaces have been changed to avoid naming conflicts when used alongside older versions of the libraries. This was a fundamental requirement to provide the ability to make considerable core changes to the extension.

- `Rotorz.TileSystem` => `Rotorz.Tile`
- `Rotorz.TileSystem.Editor` => `Rotorz.Tile.Editor`

Brushes are now represented using specialised scriptable object assets instead of prefabs. It was decided to rename all of the brush classes to help developers locate and update their source code more effectively:

- `TileBrush` => `Brush`
- `AliasTileBrush` => `AliasBrush`
- `OrientedTileBrush` => `OrientedBrush`
- `TileBrushOrientation` => `BrushOrientation`
- `AtlasTileBrush` => `TilesetBrush`
- `EmptyTileBrush` => `EmptyBrush`
- `BasicTileBrush` (removed in favor of `OrientedTileBrush`)

New important brush classes:

- `AutotileBrush`
- `Tileset`
- `AutotileTileset`

Those who were using earlier versions of Rotorz Tile System may wonder where the `TileInstance` class has gone. This class became obsolete since version 1.1.0 and was superseded by `TileData`.

Previously the naming of properties was inconsistent which was confusing to some developers. In most (if not all cases) the naming convention of properties now follows the guidelines recommended on the MSDN website http://msdn.microsoft.com/en-us/library/vstudio/ms229043(v=vs.100).aspx. For some custom scripters this change will require some minor tweaks in their scripts.

Remove Previous Version

The older version of Rotorz Tile System can be removed from your project once you have finished upgrading existing brushes and tile systems.

CAUTION

Please be careful not to remove assets that are still in use. You may want to create a second backup at this point just in case you inadvertently remove assets that are still needed.

Remove unwanted brush assets

You will need to first relocate any assets that have been manually placed within the tile prefabs folder (scripts, materials, textures, etc). This will not apply to the majority of users, however we are aware that some users have organised their assets in this way.

Remove the following assets:

```
Assets/TilePrefabs
```

Remove brush mapping asset

This asset is generated when upgrading existing brushes and is used when upgrading existing tile systems. This asset is no longer needed once you have finished upgrading your project.

Remove the following asset:

```
Assets/TileBrushes/BrushMappingsV1toV2.asset
```

Remove previous version of Rotorz Tile System

Remove the following assets:

```
Assets/RotorzTileSystem
```