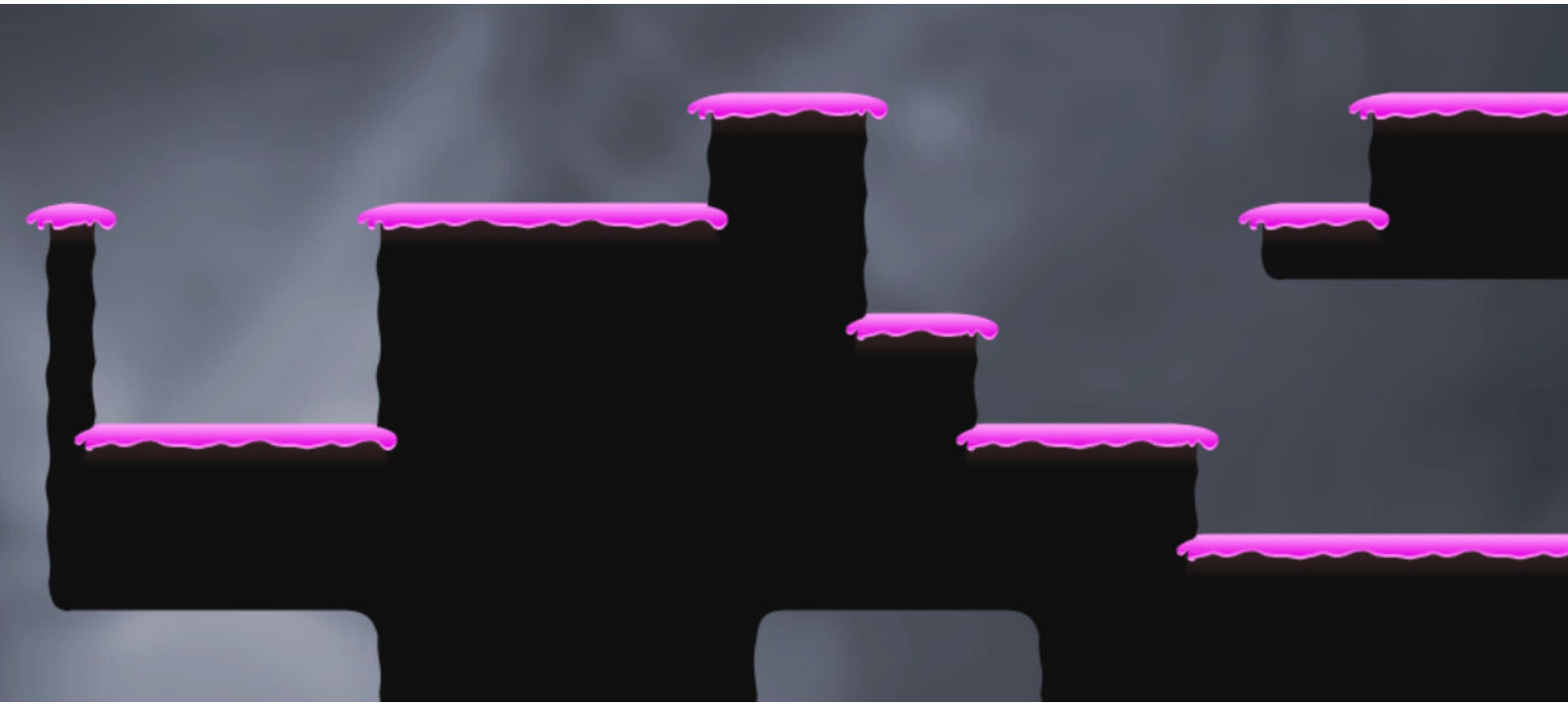


Rotorz Tile System

Editor Extension for Unity



User Guide

Table of Contents

Chapter 1. Getting Started	1
Key Features	1
Installation	2
Project Workflow	3
Technical Support	5
Updating Extension	5
Chapter 2. User Interface	6
Tool Palette	6
Tool Buttons	6
Tool Menu	6
Context Menu	8
Scene Palette	8
Tile System Visibility	8
Reordering Tile Systems	9
Locking Tile Systems	9
Renaming a Tile System	9
Brush Palette	9
List View / Tilesset	10
List Presentation	10
Filter Menu	11
Searching Brush Names	12
Keyboard Shortcuts	13
Designer Window	13
Brush Designer	13
Tilesset Designer	14
Brush Creator Tab	15
Modify Tilesset Tab	17
Info Tab	18
Lookup Tilesset Brush	19
Showing Brush in Designer	20
Showing Tilesset in Designer	21
Locking the Designer	21
Selection History	22
Keyboard Shortcuts	22
User Preferences	22
Chapter 3. Tile Systems	26
Chunks	26
Changing the Chunk Size	27
Creating a Tile System	28

Presets	29
Creating a Preset	29
Modifying a Preset	30
Deleting a Preset	31
Position, Rotate and Scale	32
Using the Inspector	32
Changing Cell Size	33
Repairing a Tile System	34
Clearing a Tile System	35
Optimization	36
Vertex Snapping	36
Smoothing	37
Stripping	37
Stripping Presets	37
Stripping Options	38
Build Options	39
Runtime Options	41
Building Prefab from Tile System	42
Building Entire Scene	43
Chapter 4. Painting	45
Tools	45
Common Tool Options	45
Deactivate Tool	47
Paint Tool	48
Cycle Tool	48
Fill Tool	49
Picker Tool	49
Line Tool	49
Rectangle Tool	50
Spray Tool	51
Plop Tool	52
Switching Between Tile Systems	55
Keyboard Shortcuts	55
Status Panel	55
Tweaking Tiles	58
Replace by Brush	58
Refreshing a Single Tile	59
Refreshing all Painted Tiles	60
Chapter 5. Brushes	61
Designing Brushes	61
Brush Categories	61

Adding a Brush Category	61
Removing a Brush Category	62
Common Brush Properties	62
Brush Name	62
Static	62
Smooth	63
Visibility	63
Tag and Layer	63
Category	63
Extended Properties	63
Group No.	64
Force Legacy Sideways	64
Scale Mode	64
Always Add Container	64
Flags	65
Offset, Rotate and Scale Prefabs	65
Remapping Materials	66
Oriented Brushes	67
What is an Orientation?	67
Variation Weights	68
Coalesce Mode	69
Fallback Orientation	71
Creating an Oriented Brush	71
Define or Find Orientation	72
Set Default Orientation	73
Remove Orientation	74
Add Variation	74
Remove Variation	75
Looking up a Variation	75
Reorder Variation	76
Copy Variation	76
Alias Brushes	77
Overriding Properties	77
Creating an Alias Brush	78
Empty Brushes	80
Creating an Empty Brush	80
Master Brushes	81
Creating a Master Brush	81
Hiding a Brush	82
Duplicating a Brush	82
Deleting a Brush	84

Chapter 6. Tilesets	86
Edge Correction	86
None	87
Borders	88
Delta / UV Inset	89
Procedural and Non-Procedural	89
Procedural Tiles	89
Non-Procedural Tiles	90
Default Procedural Setting	90
Working with Tilesets	91
Creating a Tileset	91
Locating Atlas Material	93
Changing Atlas Texture	94
Cleanup Unused Meshes	94
Deleting a Tileset	95
Tileset Brushes	96
Tileset Brush Properties	97
Procedural	97
Attach Prefab	97
Add Collider	97
Creating Tileset Brushes	97
Autotile Tilesets	98
Autotile Artwork	99
Basic Layout	100
Extended Layout	100
Autotile Expansion	100
Working with Autotile Tilesets	101
Creating an Autotile Tileset	101
Regenerate from Autotile Artwork	103
Modifying an Autotile Tileset	103
Autotile Brushes	104
Autotile Brush Properties	104
Creating an Autotile Brush	105
Chapter 7. Runtime Scripting	106
Consequences of Stripping	106
Creating a Tile System Dynamically	106
Painting at Runtime	108
Accessing Tiles	111
Chapter 8. Editor Scripting	114
Custom Tools	114
Chapter 9. FAQ	115

Chapter 10. Troubleshooting	117
Project contains redundant preview assets	117
Unable to paint on tile system	117
Changes not reflected when brush is modified	119
Chunk size too large for procedural mesh	120
Build error "count <= std::numeric_limits<UInt16>::max();"	120
Actor passes through inner tiles of smooth platform	121
Tiles not positioned, rotated or scaled as expected	122
Editor performs slowly with lots of tiles	123
Visual "glitches" between tiles	124
Poor texture alignment in tileset brushes	125
Transparency not working with tileset brushes	126
Updated tileset not reflected	126
Brush asset appears to be missing or broken	127
Glossary	i
Index	iii

Chapter 1.

Getting Started

Rotorz Tile System is an editor extension for Unity allowing you to paint maps using 2D and 3D tiles which can be great for both prototyping ideas as well as for creating fully featured games. Tiles are painted onto tile systems using brush assets which help to define the style and behavior of painted tiles.

This guide includes some information regarding runtime and editor scripting though is primarily aimed towards usage of the Rotorz Tile System editor extension itself. Further details regarding the runtime and editor APIs can be found in the [API Reference](#).



Important

Keep regular backups of your files regardless of which extensions you are using. Always be sure to review release notes and backup all relevant files before installing or updating any assets including Rotorz Tile System.

Key Features

Rotorz Tile System is a powerful and flexible extension which can be used to create a wide range of games including platformers, dungeon crawlers, mazes, puzzles and even first person shooters!

- Create brushes using a powerful but intuitive user interface.
- Effortlessly paint tiles using oriented and autotile brushes.
- Range of tools for painting, cycling, filling, picking and plopping tiles.
- Quickly customize existing brushes by creating alias brushes.
- Combine tiles into chunks to improve performance at runtime.
- Reduce number of colliders by combining adjacent box colliders.
- Utilize automatic vertex snapping to avoid visual artifacts.
- Work with reasonably large grids before experiencing poor performance.
- Edge correction to counteract edge bleeding with 2D tiles.

Installation

Rotorz Tile System can be installed by importing its package using the Unity asset store interface (or alternatively when first creating your project).

**Tip**

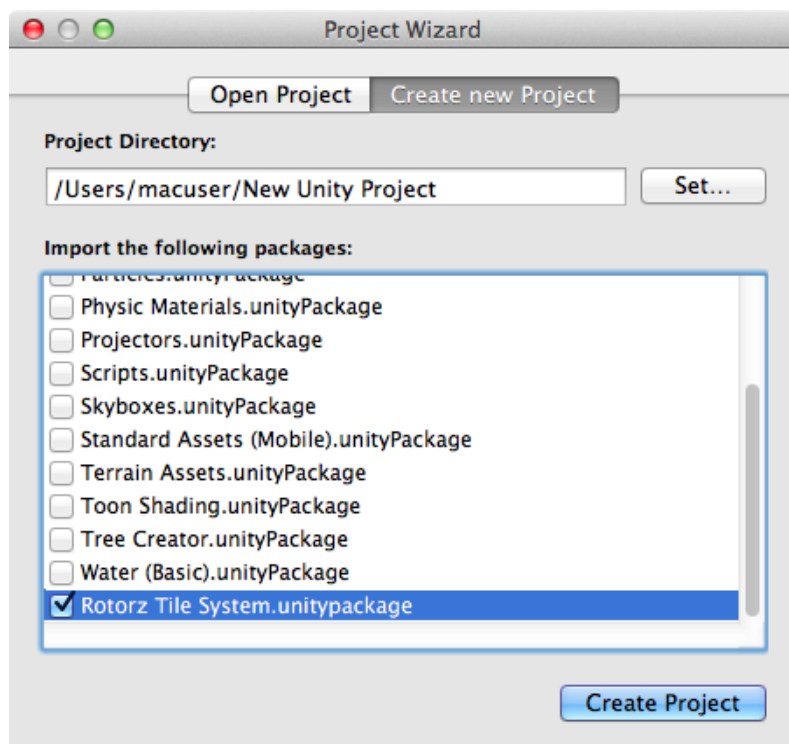
Demonstration assets are imported by default, though can be excluded when not wanted by deselecting the "Rotorz/Tile System/Demo" folder.

Import Using Asset Store

1. Open asset store **Window | Asset Store**.
2. Expand list of licensed assets.
3. Navigate to Rotorz Tile System.
4. Select **Import**.

Import When Creating Project

Package can also be selected when first creating a project:



Project Workflow

Once you've installed Rotorz Tile System into your project you can open the tool palette using the menu **Window | Rotorz Tile System**. This interface is the main entrance point to this extension and provides access to various commands and other interfaces.

Accessing documentation

The latest version of this user guide and API reference can be accessed from our website. A copy of each is also included within the support folder of the Rotorz Tile System package. The user guide is provided in PDF format and the API reference is provided in CHM format.




Note

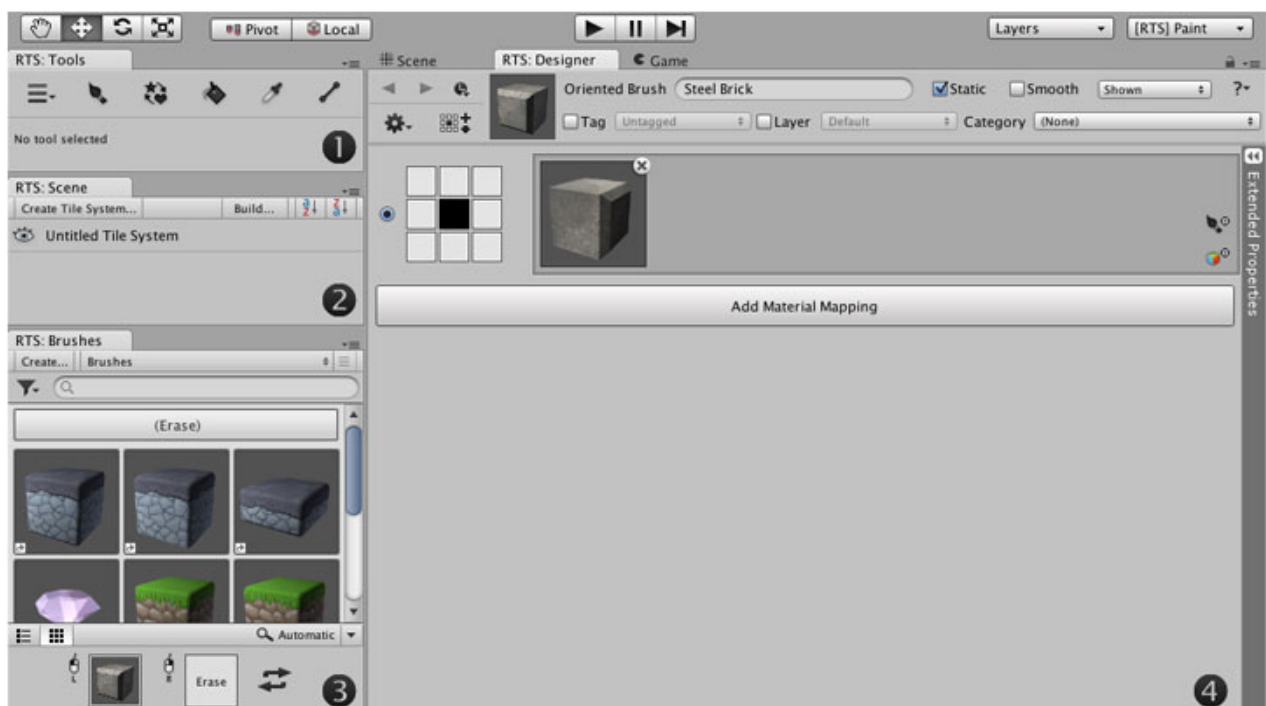
You may need to download additional third-party software before you are able to view the included PDF and CHM documentation files.

Setting up the user interface

With many thanks to customer feedback we have opted to separate our interface into multiple dockable windows allowing you to customize your editor environment to better meet your workflow. It is generally useful to dock the various palette windows into the main Unity interface, though this is entirely a matter of preference.

The following interfaces can be accessed via the main tool menu  | **Editor Windows**:

1. **Tool Palette** on page 6 - Includes tools for interacting with tile systems.
2. **Scene Palette** on page 8 - Whilst not essential, useful when working with multiple tile systems.
3. **Brush Palette** on page 9 - Lists brushes and allows selection for painting.
4. **Designer** on page 13 - Allows you to modify the active brush or tileset.



**Tip**

The **Tool** and **Brush** interfaces can be joined into a single palette window by modifying [User Preferences on page 22](#).

Setting up your scene

Tiles are painted onto a special kind of object called a [tile system on page 26](#) which must be present before you can begin to paint tiles. You can add one or more tile systems into your scene which can be positioned as needed (see [Creating a Tile System on page 28](#)).

Here are two common reasons to use multiple tile systems:

- Stack multiple tile systems to create layers of tiles.
- Define multiple grid sizes (i.e. large tiles and small tiles).

Painting tiles onto a tile system

1. With your tile system selected, activate the [Paint on page 48](#) tool.
2. Select the brush that you would like to paint with by left clicking it in palette.
3. Use left mouse button to paint tiles onto the selected tile system.
4. Drag mouse pointer whilst holding left mouse button to paint a line of tiles.

Once you have finished painting you can deactivate the selected tool by re-clicking the associated tool button, or by selecting one of the standard Unity tools.

Creating and designing new brushes

There are several kinds of brush which can be created allowing you to paint different kinds of tile. Prefabs can be added to oriented brushes allowing you to paint 3D tiles with custom behavior scripts attached. Tileset brushes can be created allowing you to paint 2D tiles; prefabs can be attached to painted tiles when desired.

An existing brush can be opened into the designer by right-clicking on it using the **Brush** palette and then selecting **Show in Designer...** from the context menu.

See [Chapter 5. Brushes on page 61](#) for further information.

Optimizing the tile systems in your scene

Tile meshes and colliders can be merged and unwanted aspects of tile systems can be stripped from final project builds by optimizing your tile systems. The way in which each tile system is optimized can be configured using the inspector.

The **Build Scene** command allows you to quickly optimize all of the tile systems in the current scene which is then saved to a separate *built* version of scene. It is generally a good idea to avoid making changes to built version of scene since such changes will be lost if original version of scene is built again.

See [Optimization on page 36](#) for further information.

Technical Support

If you experience difficulties getting Rotorz Tile System to run, the first thing that we recommend is to read through [Frequently Asked Questions](#) on page 115 and review the [Troubleshooting Guide](#) on page 117 to see if there is already an answer to your problem.

Use our [forums](#) for questions, discussions and to show off what you are working on!

You can contact us via our website: <http://rotorz.com/contact>

Updating Extension

Asset store customers can update Rotorz Tile System via the Unity asset store. Always backup your files and read through release notes before updating to the latest version!

**Tip**

Follow [@rotorzlimited](#) on Twitter or [Like rotorzlimited](#) on Facebook to learn about new updates and keep up to date with the latest.

Each update may contain new features and of course bug fixes. No software is ever perfect and Rotorz Tile System is no exception. Rotorz Limited strives to zap bugs as quickly as possible.

Rotorz Tile System is an extension for the Unity software and as such is limited by the capabilities of Unity itself. Occasionally updates to Rotorz Tile System are necessary to workaround changes that have been made to the API exposed by Unity; often to take advantage of yummy new features!

We strive to keep the update process as simple as possible, though manual steps are sometimes necessary to take full advantage of the latest update. There is currently no reliable way to automate such changes upon importing the latest version of our extension. In many respects this is good since it gives you greater control over the way in which you update your project.

Chapter 2.

User Interface

Rotorz Tile System adds a selection of user interfaces and functions to the Unity editor which aim to provide you with a familiar development experience. Palette windows provide essential functionality which can be positioned and docked to suite your workflow.

Tool Palette

The tool palette is the main entry point for Rotorz Tile System and contains tools which can be used to interact with tile systems. Additional Rotorz Tile System features can be accessed via the main tool menu.

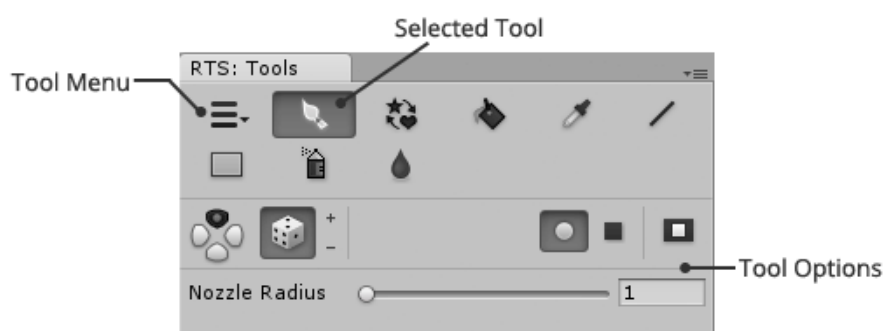


Figure 2-1. Tool palette interface

This window can be accessed via the menu **Window | Rotorz Tile System**.

Tool Buttons

A number of tools are provided allowing you to interact with tile systems in various ways. Visibility and ordering of tool buttons can be adjusted using the preferences window.

Tools can only be selected when your scene contains at least one tile system. The first tile system in your scene will become active when one is not already active. Selected tool can then be used to interact with the active tile system.

Additional options relating to selected tool are shown below when available. For more information regarding each of the provided tools and their accompanying options refer to [Tools on page 45](#).

☰ Tool Menu

Displays a popup menu allowing you to access the various features offered by Rotorz Tile System including the various palette windows.

**Note**

Prior to Rotorz Tile System version 2.1.0 a similar "RTS" menu was available from the main Unity menu bar. Please refer to the 2.1.0 release notes for further information regarding this change.

See below for summary of each menu item:

Menu Command	Description
Create Tile System...	Create a new tile system. See Creating a Tile System on page 28 .
Create Brush or Tileset...	Create a new brush or tileset. See Chapter 5. Brushes on page 61 .
Use as Prefab Offset	Use manual alignment, rotation and scale offsets of selected tile object as prefab offset so that such tiles are aligned correctly in the future. See Offset, Rotate and Scale Prefabs on page 65 .
Replace by Brush...	Find tiles that were painted using a specific brush and repaint them using another brush. Find and replace for tiles! See Replace by Brush on page 58 .
Build Scene...	Optimize all tile systems in scene using per tile system properties and then save as separate optimized scene. See Building Entire Scene on page 43 .
Rescan Brushes	Scan project for new or missing brushes; useful for occasions where brushes have been imported from an asset package.
Editor Windows Designer	Shows designer for selected brush or tileset. See Designer Window on page 13 .
Editor Windows Scene	Shows list of editable tile systems in current scene and highlights the active one. See Scene Palette on page 8 .
Editor Windows Brushes	Palette window allowing you to view and filter brushes. You can then paint tiles using the selected brush. See Brush Palette on page 9 .
Editor Windows Upgrader	A wizard which assists you when upgrading from older versions of Rotorz Tile System. Refer to associated migration guide for further information.
Online Resources	Various links to online resources including this user guide, the API reference and the Rotorz Tile System home page!
Preferences...	Use to tweak various preferences including the coloring of tile system grids. See User Preferences on page 22 .

Menu Command	Description
About...	Information about Rotorz Tile System including the version number and a link to our Twitter profile!

☰ Context Menu

The tool palette window adds a command to the Unity editor window context menu allowing you to reset tool options.

See below for summary of each menu item:

Menu Command	Description
Reset Tool Options	Reset options of each tool to their default values. This command does not reset user preferences (see User Preferences on page 22).

Scene Palette

Displays list of editable tile systems in the current scene allowing quick selection of the active tile system. Tile systems can be shown, hidden, renamed and rearranged using this interface.

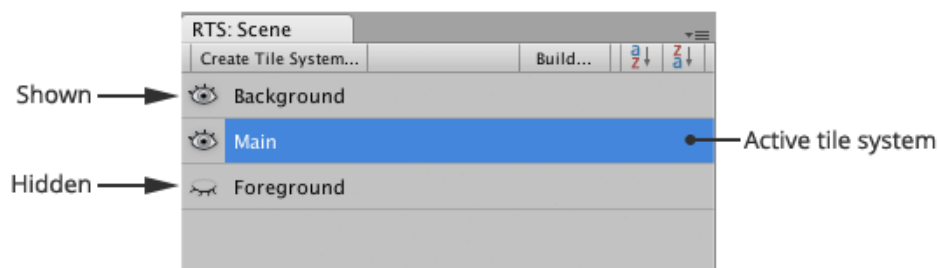


Figure 2-2. Scene palette window

This interface can be displayed via the tool menu ☰ | **Editor Windows** | **Scene**.

The labels of tile systems can be dragged and dropped onto other user interfaces which is useful when associating tile systems with custom scripts.

Right click on tile system to reveal handy context menu.

Tile System Visibility

Tile systems can be temporarily hidden to reveal obscured tiles and objects in your scene by clicking the toggle icon.

- - Shown when tile system is visible, click to hide tile system.
- - Shown when tile system is hidden, click to reveal tile system.





Important

This functionality is only available for Unity 4 users.

Reordering Tile Systems

Tile systems can be manually ordered by dragging them within the scene palette which is particularly useful when you have multiple layers of tile systems in your scene.


- Tile systems can be dragged and dropped within scene palette.
- Click  (Sort Ascending) to sort list of tile systems by name in ascending order.
- Click  (Sort Descending) to sort list of tile systems by name in descending order.

Ordering of tile systems does not affect the order in which they are rendered in your game. To change the visual ordering of tile systems you must adjust the position of your tile system using the move tool (or alternatively using the transform inspector).

Locking Tile Systems

Tile systems can be locked to avoid inadvertent interaction with the active paint tool.

It is not possible to paint or erase tiles on locked tile systems and most actions are disabled to prevent accidental alterations. Placeholder messages are shown within scene view and inspector indicating that tile system must be unlocked before changes can be made.

Tile system can be locked or unlocked by context clicking tile system and then toggling **Lock**. A small padlock icon  is shown to denote that a tile system has been locked.

Renaming a Tile System

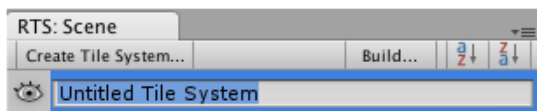
You can rename tile systems using the scene palette in much the same way as you would with the standard hierarchy window.

Procedure

1. Right-click on the tile system that you would like to rename.
2. Select **Rename** from context menu.

Rename field can also be shown using the hot key **F2** (Windows) or **Return** (OS X) when scene palette is focused.

You should then see something like the following:



3. Input new name for tile system.
4. Press **Return** key to accept new name, or press **Escape** key to cancel.

Brush Palette

Displays list of brushes allowing selection of primary and secondary brush for use with selected tool. Primary brush selection can be edited when designer window is shown.

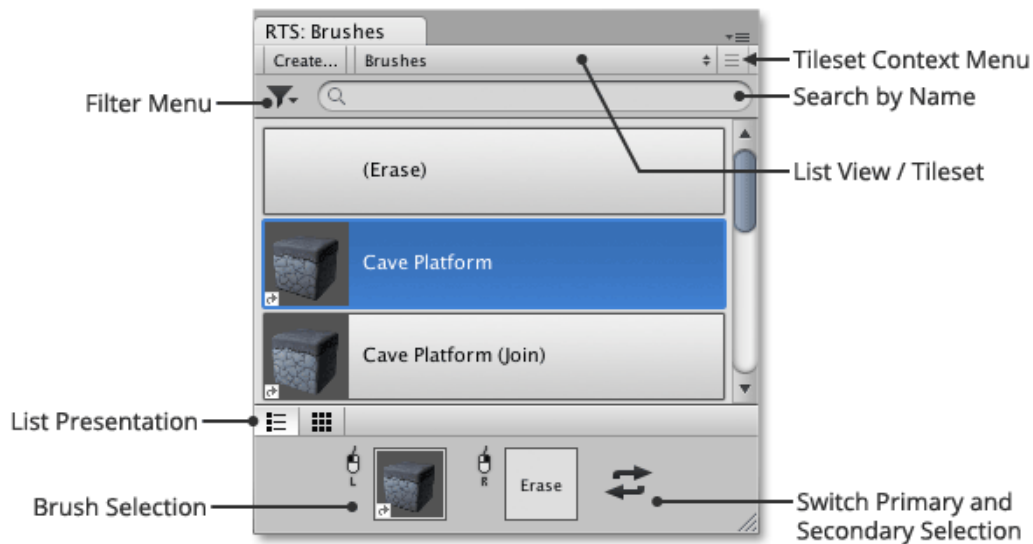



Figure 2-3. Brush palette window

This interface can be displayed via the tool menu  | **Editor Windows** | **Brushes**.

List View / Tileset

Allows you to specify whether general brushes or tileset brushes are to be viewed in the brushes list. For convenience **Erase** is available from all views.

The brush palette contains two types of view:

- **Brushes** - General list of brushes where tileset brushes are usually hidden (unless favored or **Hide Tileset Brushes** is deselected from filter menu).
- **Tileset** - Brushes from a specific tileset.

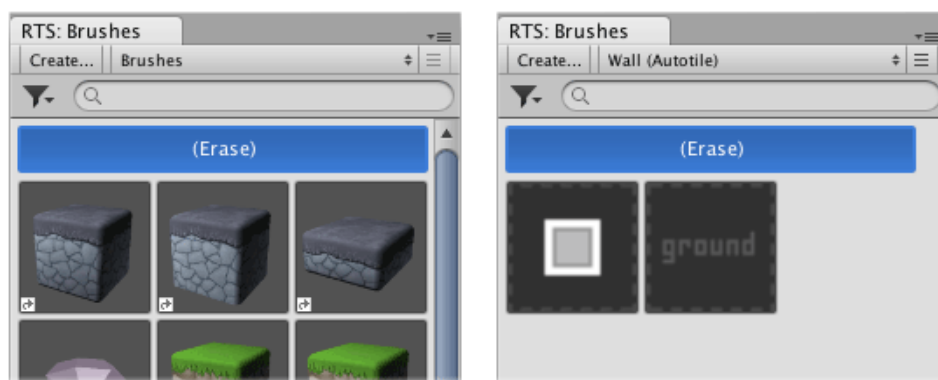


Figure 2-4. Brush list viewing modes



Tip

Brushes that are not intended to be used directly can be hidden using the brush designer. Master brushes are not shown in the brush palette.

List Presentation

Brushes can be shown as a simple list of labelled thumbnails or alternatively as a gallery of icons. Presentation can be selected using buttons located at lower left of brush list.

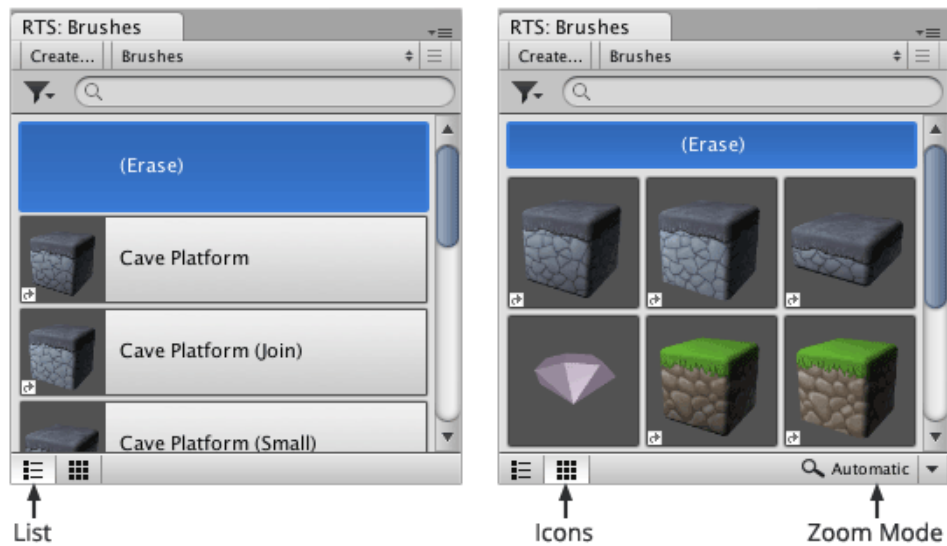


Figure 2-5. Brush list presentation modes

**Tip**

Hover mouse pointer over brush to reveal its name for icons presentation.

Zoom Mode

Allows you to control how brush thumbnails are scaled when icons presentation is used by selecting one of the following modes:

- **Automatic** (Default) - Use tile size when a tileset is active, otherwise adjust zoom to make use of the available space.
- **Best Fit** - Adjust zoom to make use of the available space.
- **Custom** - Displays slider allowing user to manually adjust zoom.

Filter Menu

The filter menu contains a selection of options allowing you to control the way in which brushes are filtered. Brushes can also be organized into categories which can then be quickly filtered.

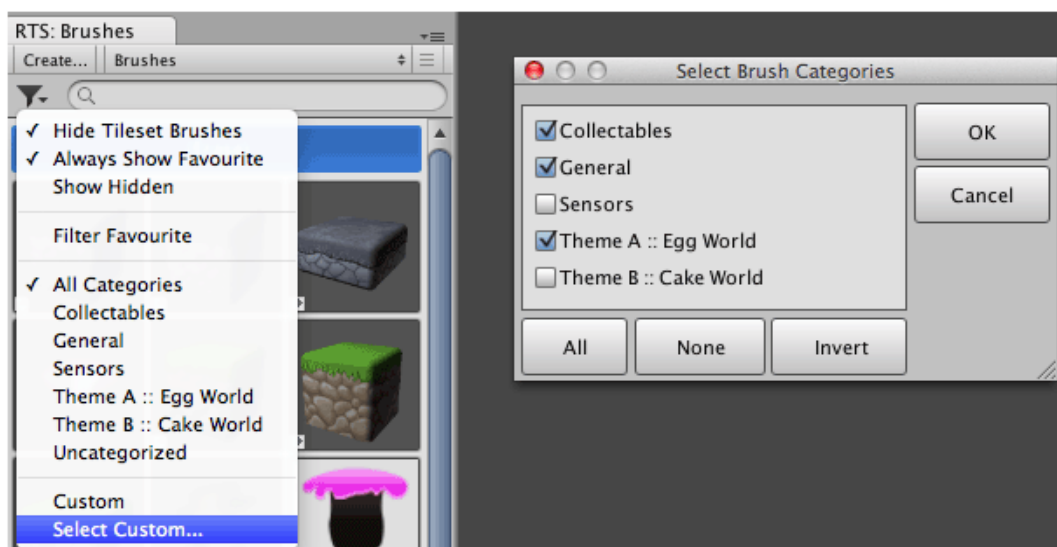


Figure 2-6. Filter brushes using categories

Filter Options

Upper part of filter menu contains the following options:

- **Hide Tileset Brushes** - Tileset brushes are only shown when a tileset is selected by default, though they can be shown in other views by deselecting this option.
- **Always Show Favorite** - Tileset brushes are hidden from the brushes view when the prior option is selected. However, this option allows you to force favorited tileset brushes to be shown regardless.
- **Show Hidden** - Occasionally brushes are hidden so that they do not appear in the brush palette; usually when they are not intended for direct usage. This option allows you to temporarily reveal hidden brushes so that you can edit them using the designer.
- **Filter Favorite** - Only favorited brushes will be shown when selected.

Filtering by Category

Lower part of filter menu allows brushes to be filtered by custom categories that are defined using the brush designer. Category filtering options will only appear when at least one brush category has been defined:

- **All Categories** - Shows brushes from all categories.
- **Uncategorized** - Shows brushes which have not been categorized.
- Select category name to show brushes from that category.
- **Select Custom...** - A custom pick of categories.
- **Custom** - Restores previous custom category selection.

Clear Active Filter

To clear filter simply click "X" button as illustrated below:

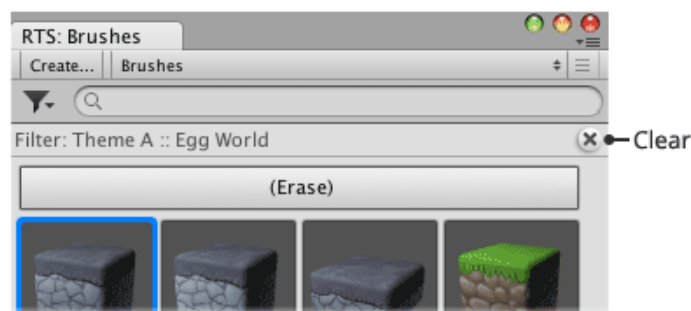


Figure 2-7. Clearing brush list filter

Related information

- [Brush Categories on page 61](#)

Searching Brush Names

The search field allows brushes to be filtered by name. When specified only brushes whose name contains the search text will be shown. Search can be cleared by clicking the small cross button.

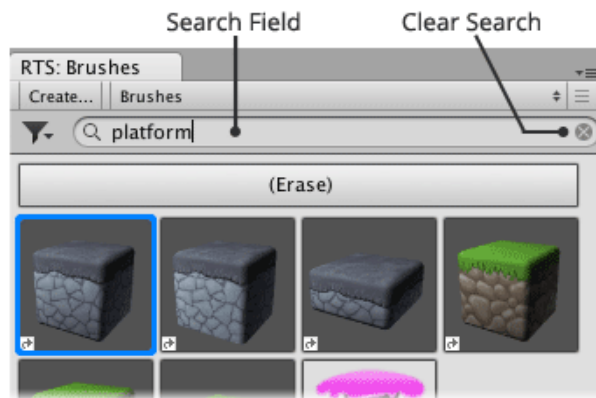


Figure 2-8. Filter brushes by name with search text



Tip

This feature can be used in conjunction with category filtering to make it even easier to find the brushes that you need.

Keyboard Shortcuts

The brush palette includes two keyboard shortcuts allowing you to quickly set the primary or secondary brush and begins painting if tool is not already selected.

- **Ctrl + Left Click** - Set primary brush and begin painting.
- **Ctrl + Right Click** - Set secondary brush and begin painting.

Designer Window

The designer window is a central interface which can be used to modify the active brush or tileset. Each type of designable object comes with its own specialized interface which can be quickly accessed via the brush palette.

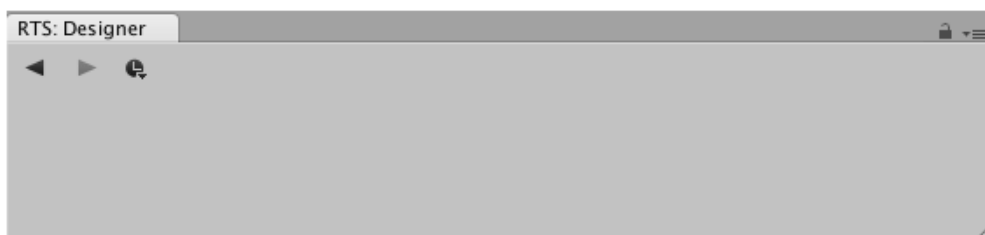


Figure 2-9. Designer window

This interface can be displayed via the tool menu  | **Editor Windows** | **Designer**.

Brush Designer

Each type of brush has its own specialized designer interface exposing relevant commands and properties. Common properties are shown along the top of the interface whilst more advanced options are tucked away until needed.

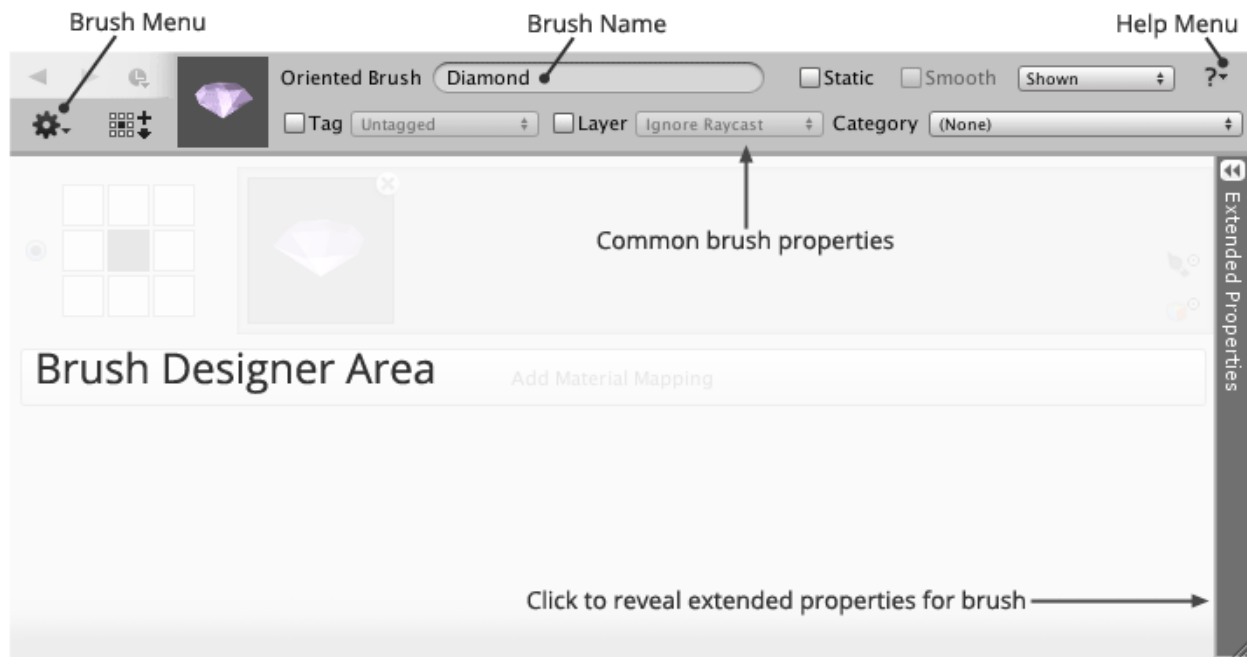


Figure 2-10. Designer window with oriented brush selected

Common Properties

Properties common to most brush types which are used frequently.

Brush Designer Area

Provides the controls needed to modify the selected brush. Some designers are broken down into further sections which can be toggled.

Extended Properties

Advanced or less commonly used options which are hidden by default which can be toggled by clicking the caption area.

Related information

- [Chapter 5. Brushes on page 61](#)

Tileset Designer

Tilesets can be modified using the designer window which can be accessed via the brush palette. Tileset brushes can be created using this interface.

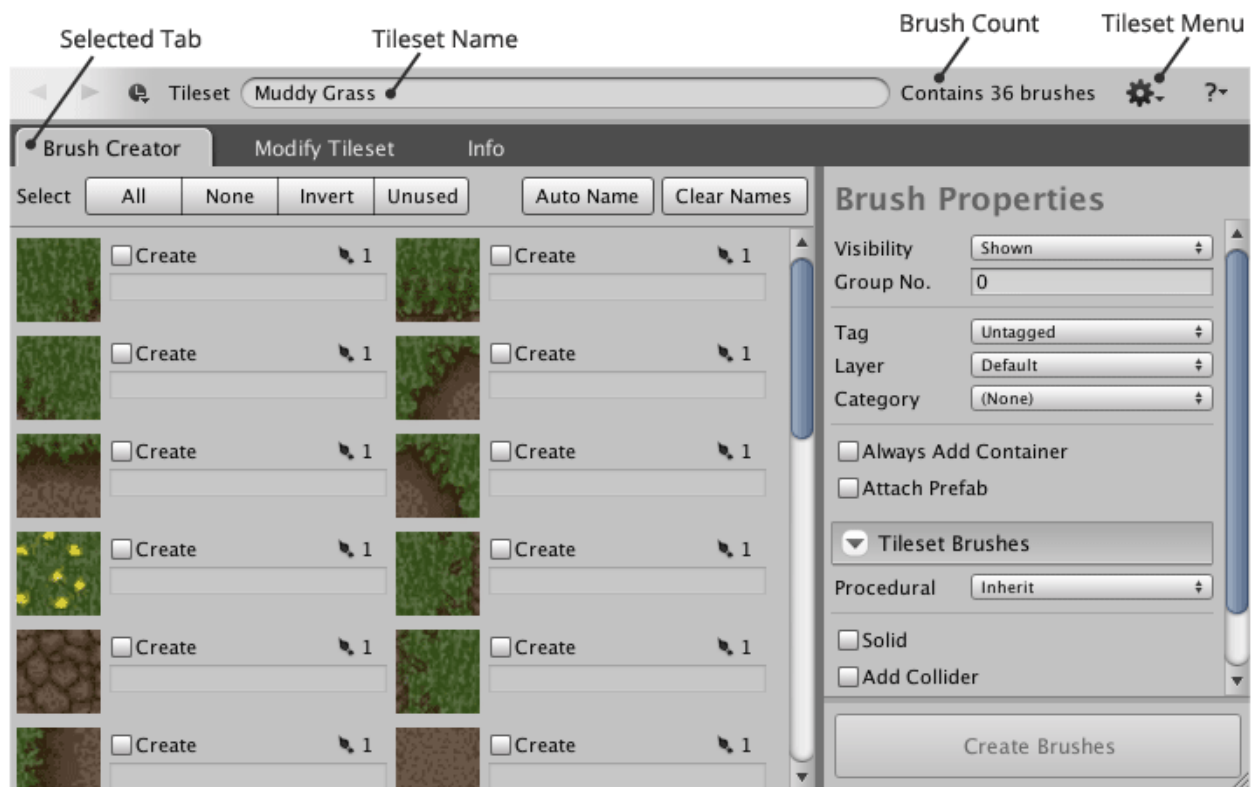


Figure 2-11. Designer window with tileset selected

Related information

- [Chapter 6. Tilesets on page 86](#)
- [Working with Tilesets on page 91](#)

Brush Creator Tab

Tileset brushes can be created using the brush creation interface of the tileset designer. Brushes can be created in batches by selected multiple tiles.

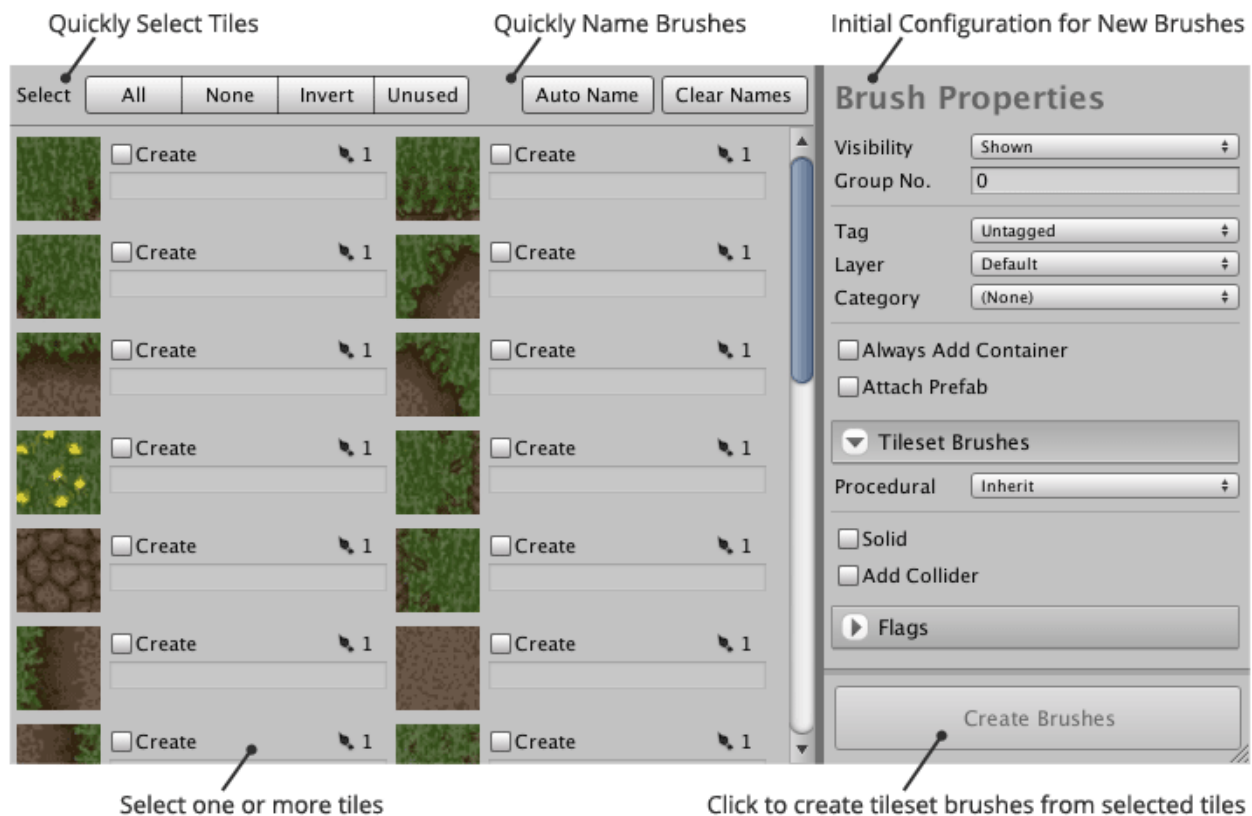


Figure 2-12. Brush creator tab of tileset designer

Related information

- [Creating Tileset Brushes on page 97](#)

Tiles

Each available tile in the active tileset is listed on the left side of the brush creator interface. The following interface elements are presented for each tile.

Tile Thumbnail

Thumbnails exclude additional border space making them easier to understand. Tile previews are stretched to fit the thumbnail area which may cause them to appear distorted.

Create Toggle Field

Selecting this field indicates that a new brush should be created to represent the tile. This makes it easier for you to create brushes for a selection of tiles.

Brush Name

Allows you to specify the name will be assigned to the newly created brush. Selecting this field will automatically select the **Create** toggle field.

An error message will be displayed directly below this field if a naming conflict or input error is detected.

Count of Brushes for Tile

Displays the total number of brushes that have been created for tile.

Clicking this button will navigate to the brush. A popup menu is displayed when there is more than one brush for tile allowing you to pick the specific instance that you would like to navigate to.

Tile Selection

A series of buttons are displayed immediately above the list of available tiles which allow you to quickly select the tiles that you would like to create brushes for.

All

Select all tiles.

None

Deselect tiles.

Invert

Invert the current selection of tiles.

Unused

Select tiles that do not have an associated brush.

Naming Brushes

These buttons allow you to automatically populate the brush name field of each tile where **Create** has been selected.

Auto Name

Assigns a unique name for each tile.

Clear Names

Erase name field for each tile.

Brush Properties

When creating a batch of brushes it is often useful to assign the same properties to each newly created brush. This panel allows you to do just that!

The properties presented at the top of this panel will be applied to both tileset and autotile brushes upon creation. The remaining properties have been grouped into foldout sections:

Tileset Brushes

Properties that only apply to regular tileset brushes.



Note

These properties are ignored when creating autotile brushes.

Autotile Brushes

Properties that only apply to autotile brushes. This section is displayed when an autotile brush has been selected for creation.

Flags

Flags that will be used when painting tiles. Flags will be assigned to each created brush.

Modify Tileset Tab

Allows you to modify to existing tilesets. In the case of autotile tilesets it is also possible to regenerate the atlas texture from the input autotile artwork.

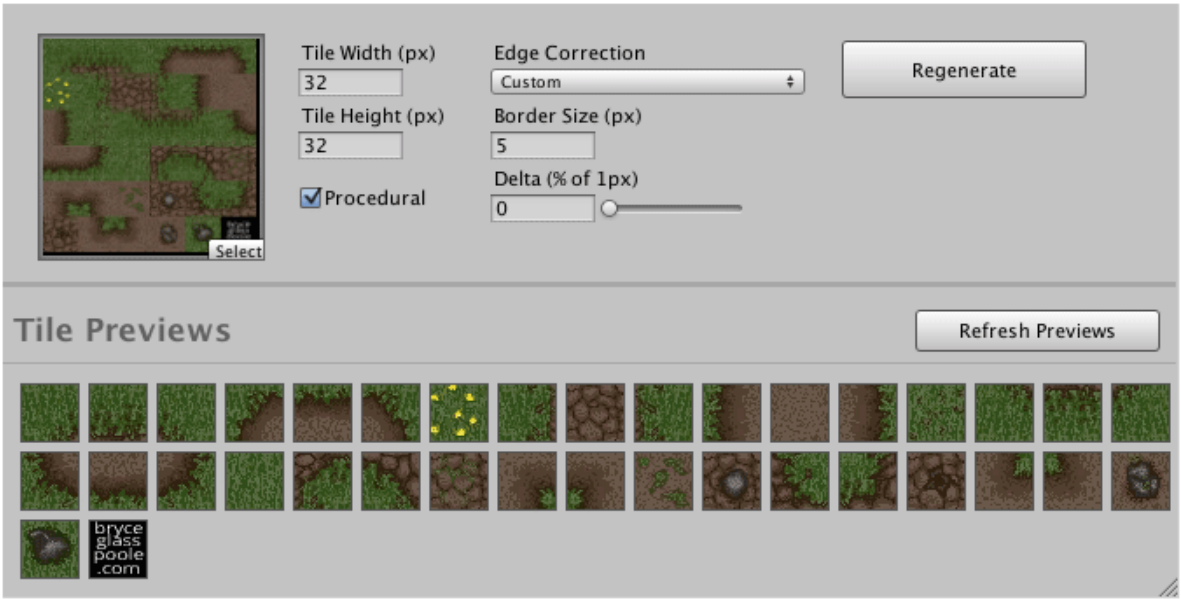


Figure 2-13. Modify tab of tileset designer

Tile previews are rendered for the specified tileset at actual size. Some modifications require additional processing before previews can be displayed. In such cases the **Refresh Previews** button can be clicked. For performance reasons the maximum number of tile previews displayed is constrained.

Related information

- [Changing Atlas Texture on page 94](#)
- [Regenerate from Autotile Artwork on page 103](#)

Info Tab

Provides information about active tileset along with preview of atlas texture. Additional metrics are shown to right of atlas preview.



Figure 2-14. Information tab of tileset designer

Path of tileset assets is displayed at top of interface which is relative to the project folder. Tilesets usually comprise of a number of assets (tileset asset, material, autotile atlas texture) which are grouped into folders for convenience.

Click **Show in Finder...** (OS X) or **Show in Explorer...** (Windows) to reveal contents of folder.



Note

Tileset brushes are stored inside their respective tileset assets.

Atlas / Autotile Atlas

Layout (autotile only)

Indicates whether autotile atlas was generated from a basic or extended layout.

Inner Joins (autotile only)

Indicates whether autotile atlas includes the artwork for inner join tiles.

Type

Indicates whether tileset brushes are procedural or non-procedural by default. This option can be overridden by individual tileset brushes.

Width, Height

Size of atlas texture in pixels. The reflected size may differ from the actual size of the texture image according to the way its import settings are specified within Unity.

Rows, Columns

Maximum number of rows and columns of tiles within atlas.

Tile Size

Size of single tile in pixels excluding any additional border size.

Edge Correction

Border

Border size in pixels that surrounds each tile. The offset between tiles can be calculated as:

```
Horizontal Offset = Tile Width  + ( Border Size × 2 )
Vertical Offset   = Tile Height + ( Border Size × 2 )
```

Delta

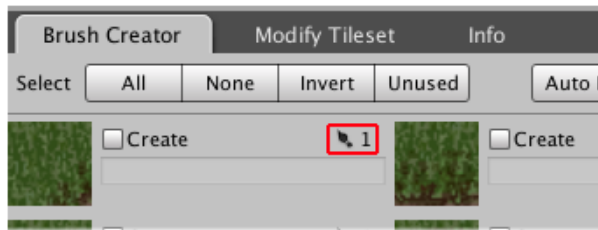
UV coordinates can be inset by a small amount to avoid visual artifacts when viewed. The UV delta specifies the percentage of a pixel to indent by.

Lookup Tileset Brush

The brush creator tab includes a handy shortcut for quickly navigating to a specific brush instance for any available tile.

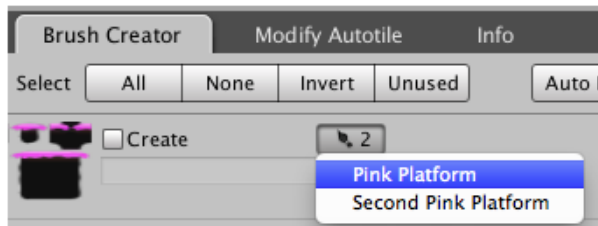
Just one brush for tile:

Clicking the brush counter will immediately navigate to the brush that is associated with the tile:



Multiple brushes for tile:

When there are multiple brush instances for a single tile you can pick the one that you would like to navigate to from the popup menu:



Related information

- [Working with Tilesets on page 91](#)

Showing Brush in Designer

Selected brush is automatically shown in designer when designer window is already open, otherwise can be accessed via context menu by right-clicking brush in palette.

Before you begin

Ensure brush palette is shown by selecting  | **Editor Windows** | **Brushes**

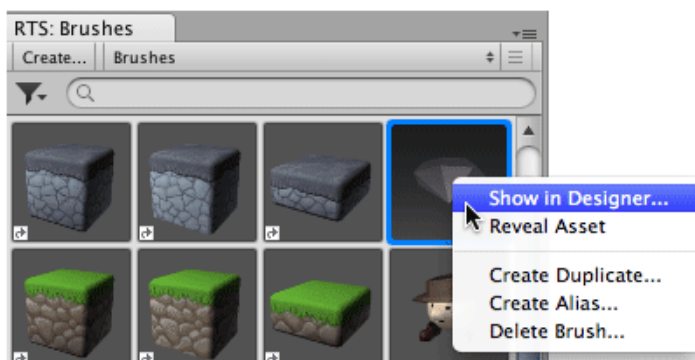


Fastpath

Shortcut: Double-click brush to quickly show designer.

Procedure

1. Right-click brush in **Brush** palette.
2. Select **Show in Designer...** from context menu.



Showing Tileset in Designer

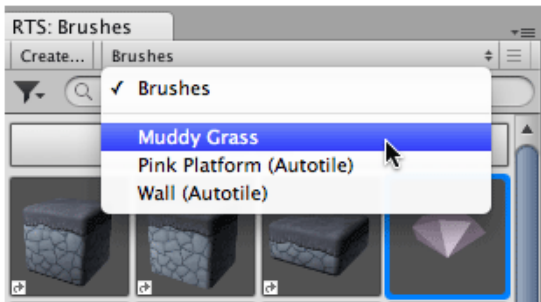
Designer for active tileset can be accessed via the tileset context menu in brush palette.

Before you begin

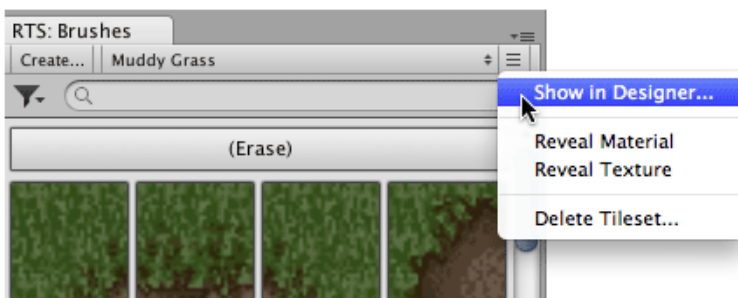
Ensure brush palette is shown by selecting  | **Editor Windows** | **Brushes**

Procedure

1. Select tileset in **Brush** palette.



2. Select menu command  | **Show in Designer...**





Locking the Designer

Designer can be locked so that active brush or tileset is retained whilst another brush or tileset is selected using the brush palette. This is similar to the inspector lock and is particularly useful when dragging and dropping brushes.



Click lock icon when brush or tileset is shown in designer to toggle:

-  - Not Locked
-  - Locked

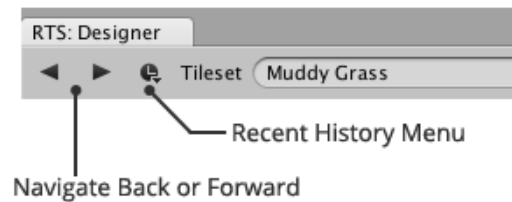


Tip

You can still navigate between designers using the selection history when designer window is locked since this is often quite useful.

Selection History

You can quickly navigate to recently selected objects using the back and forward buttons in the upper left corner of the designer. The most recently accessed objects can also be conveniently accessed via a drop-down menu.



Designer Keyboard Shortcuts

Several keyboard shortcuts have been added to help improve your workflow when using the designer window. These shortcuts will only work if the designer window is focused.

Shortcut	Description
F2	Display Create Brush / Tileset window.
F3	Display window to find or define an orientation when modifying oriented brushes.
Ctrl + E	Toggle visibility of extended properties panel.
Alt + Left Arrow	Go back to previously selected brush.
Alt + Right Arrow	Go forward to previously selected brush.
Page Up or Ctrl + Up Arrow	Select previous brush in brush palette.
Page Down or Ctrl + Down Arrow	Select next brush in brush palette.
Ctrl + Home	Select first brush in brush palette.
Ctrl + End	Select last brush in brush palette.

User Preferences

A number of preferences have been exposed allowing you to customize the appearance and functionality of Rotorz Tile System. The user preferences window also includes a button allowing you to restore the default values.

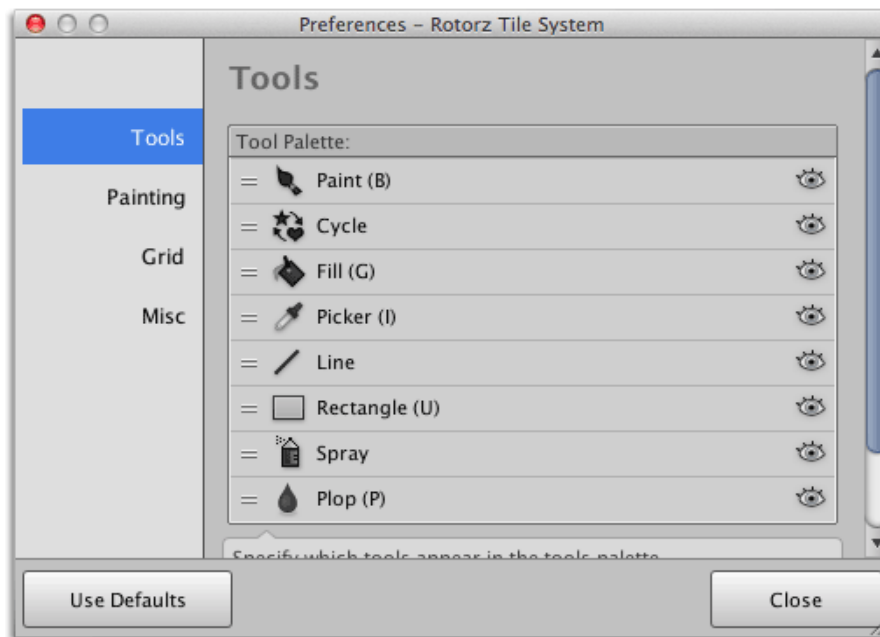




Figure 2-15. User preferences

This interface can be displayed via the tool menu  | **User Preferences...**



As you might expect, preferences are stored globally and are shared between all projects that make use of Rotorz Tile System. Preferences are stored in a JSON data file inside your application data directory which varies according to operating system:

- **Windows:** C:\Users\{UserName}\AppData\Roaming\Rotorz\Rotorz Tile System\Settings.json
- **OS X:** /Users/{UserName}/.config/Rotorz/Rotorz Tile System/Settings.json

Tools

Preference Name	Description
Tool Palette	<p>Drag listed tools to reorder them in the tools palette. You can also customize which tools are shown:</p> <ul style="list-style-type: none"> •  - Shown when tool is visible, click to hide tool. •  - Shown when tool is hidden, click to reveal tool.
Automatically show tool palette upon activating tool	Indicates whether tool palette should be shown when activating tool using hot-key or via API since tools can be used without tool palette window being shown.
Join tool and brush palettes	When active the Tool and Brush palette windows are combined into a single window offering a similar experience to the original user interface, and has been included for those who have become accustomed with the original tile system panel.


Painting

Preference Name	Description
Erase empty chunks	<p>Indicates whether chunk game objects should be removed from scene as they become empty:</p> <ul style="list-style-type: none"> • Yes (default) - Chunks should be erased when they become empty. This can cause a minor pause when chunks are repeatedly removed and then re-added as a result of painting; though it helps to avoid including unnecessary chunks in your scene. • No - Leave chunks even though they are empty. • Per Tile System - Use per tile system stripping option to determine whether empty chunks should be erased.
Preferred nozzle indicator	<p>Nozzle indicator is used to highlight location of tile that will be painted by the selected tool:</p> <ul style="list-style-type: none"> • Automatic (default) - Allow brush to determine which indicator to use. • Wireframe - Wireframe cube should be used to highlight active tile when possible. • Flat - Flat shaded tile should be used to highlight active tile when possible. <p>Color of nozzle can be adjusted by specifying a custom wireframe and shading color. Alpha component of color can be used to control opacity of nozzle indicator.</p>
Display immediate previews	<p>Shows preview prior to painting tile making it easier to visualize and align larger tiles. Preview may not always reflect exact output of some brushes and can be disabled on per brush basis if desired.</p> <p>Preview can be tinted with custom color where alpha component controls opacity of color tint.</p> <div>  <p>Note Appearance of immediate previews can be customized by adjusting the "Rotorz/Preview" or "Rotorz/Preview See Through" shaders which are located in <code>/Assets/Rotorz/Tile System/Shaders/</code>.</p> </div>
See-through previews	<p>When enabled you can see-through existing tiles to reveal preview of replacement tile.</p> <div>  <p>Note Holding the control key whilst painting will temporarily toggle the value of this preference.</p> </div>

Grid

Preference Name	Description
Colors	Customize color of tile system grid lines in scene view where alpha component controls opacity of background/ grid lines.
Highlight active tile system	Shows fainter grid for active tile system even when not actually selected for a useful visual guide when manually aligning objects.
Display grid lines	Specifies whether grid lines should be shown in scene view for selected tile systems.
Display chunk boundaries	Specifies whether chunk boundaries should be highlighted for selected tile systems.
Hide scene view grid upon activating tool	Specifies whether Unity's placement grid should be temporarily hidden in scene views when a Rotorz Tile System tool is active.

Misc

Preference Name	Description
Preferred Language (BETA)	<p>This is an early preview for an option which allows the user interfaces of Rotorz Tile System to be shown in other languages.</p> <p>The small refresh button to the side forces a reload of the current "Preferred Language" which is useful when creating custom localizations.</p>
Enable warning when creating large tile systems	By default a warning message is displayed when creating large tile systems. This warning can be suppressed by deselecting this preference.
Show detailed tips when available	Detailed tips are available in some user interfaces to provide extra information. Such interfaces usually provide an option to quickly toggle this preference on or off.
Display custom cursors	Custom cursors have been added for tools providing useful visual hints though these can be disabled to assume default arrow cursor.
Always center utility windows.	Utility windows are normally only centered upon first being shown. Enable this preference to center utility windows each time they are shown.
Check for legacy brush/tileset suffixes on startup.	<p>Prior to Unity 4.5 there wasn't an API to load all assets of a specific type so '.brush' and '.set' suffixes were added to asset names allowing them to be identified.</p> <p>Since these are no longer needed they can be automatically detected and removed. Enable this preference to display prompt if such suffixes are detected upon opening a project.</p> <p> Rescan Brushes always performs this check.</p>

Chapter 3.

Tile Systems

A tile system is a grid which allows you to effortlessly paint both 2D and 3D tiles alongside one another using brushes. Tiles are automatically aligned and oriented to their tile system upon being painted.

Tiles can be painted using the provided [tools on page 45](#) or by custom runtime or editor scripts. Such scripts typically interact with tile system components and brush assets allowing you to procedurally generate maps or even implement your very own in-game level designer!

Each tile system is a game object with the the `Rotorz.Tile.TileSystem` component attached. With the exception of tiles that are painted using purely procedural brushes, each tile is an individual game object. Data is stored on a per tile basis which can be accessed using the runtime API or stripped from builds when not needed at runtime (see [Stripping on page 37](#)).

Chunks

Tiles are mapped using a chunked data structure which helps to avoid memory wastage when large areas of a tile system remain empty. Custom scripts can access tile data transparently of the underlying data structure.

Chunks are also home to the meshes generated by procedural tileset brushes. Procedural meshes are typically generated at runtime since this helps to reduce the file size of your scenes. There is a build option which allows you to pre-generate such meshes if preferred.

Tile meshes can be combined by material on a per chunk basis helping you to reduce the total number of draw calls in your scene. This is similar to static batching and can help to improve performance, especially on mobile platforms. Tiles painted using procedural tilesets are not combined since they are effectively already combined.



Tip

Whilst tile meshes can be combined on a per-chunk basis, it is possible to specify a different chunk size for the mesh combiner if desired (see [Build Options on page 39](#)).

Please remember that there is a maximum of around 64K vertices per mesh which means that the chunk size must be chosen accordingly. For example, the maximum chunk size for a procedural mesh has an area of 100x100 tiles (though not necessarily square, it could be 10x1000).

Visualizing Chunks

Chunk boundaries of the active tile system are outlined by default. This can help you to make better use of resources when planning your game.

The visualization of chunk boundaries can be hidden if you find them distracting by deselecting **Display chunk boundaries** (see [User Preferences](#) on page 22).

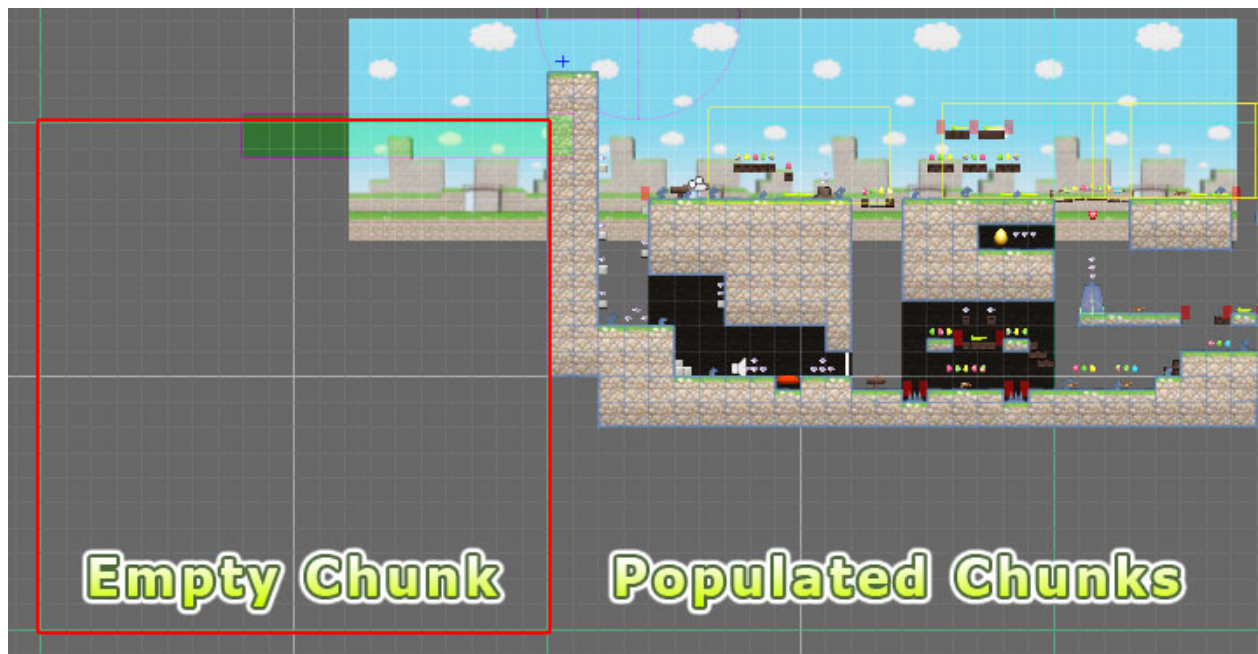


Figure 3-1. Demonstration of chunk usage in Munchy Bunny!

Changing the Chunk Size

The chunk size of a tile system can be changed using the inspector window which is useful when optimizing your scenes.

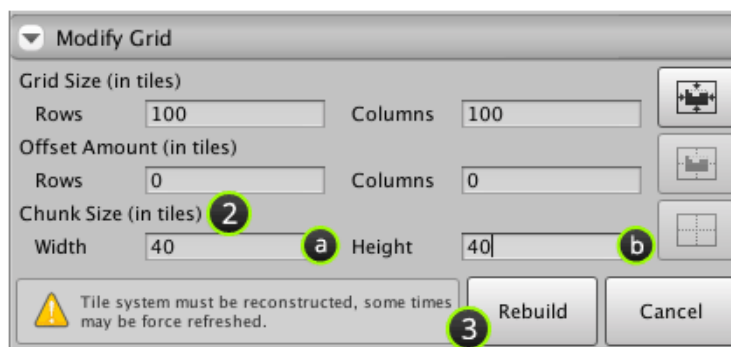
Before you begin

Ensure that **Inspector** is shown by selecting **Window | Inspector**.

Procedure

1. Select root game object of the tile system that you would like to change.
2. Input the number of rows and columns to use per chunk using **Inspector**.

Once changed a message should appear indicating that the tile system must be reconstructed using the newly specified chunk size:



CAUTION

Some tiles may be force refreshed whilst reconstructing tile system. You can abort at this stage by clicking **Cancel**.

3. Click **Rebuild**.


Related information

- [Chunk size too large for procedural mesh on page 120](#)

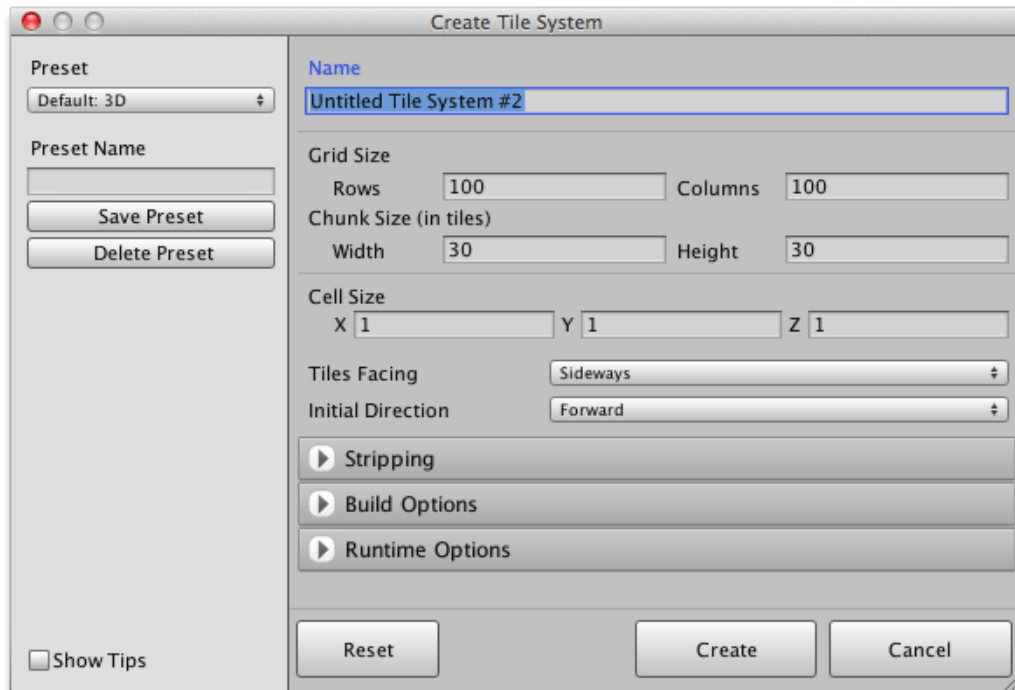
Creating a Tile System

The tile system creation interface can be accessed via the main tool menu allowing you to define the specifics for your tile system. You can later modify your tile system using the inspector if desired.

Procedure

1. Select menu command  | **Create Tile System....**
 - or, click **Create Tile System...** in toolbar of **Scene** palette.
 - or, select **Create | Rotorz Tile System...** menu in **Hierarchy** window.

The following window should appear:



2. Specify a name for your new tile system.
3. Specify the number of rows and columns of tiles in your system.



CAUTION

Larger systems will generally consume more memory even when left empty, and if too large may cause the Unity editor to crash.

4. Specify the size of a chunk.

Optimal chunk size will vary depending upon your requirements. It is often useful to experiment with different chunk sizes and then measure performance and resource utilization.



Tip

A chunk size of 100x100 often works well for 2D tile maps.

5. Specify dimensions of a tile cell in world space.

The way in which tiles are scaled to fit the specified cell size can be defined using the brush designer. It is common to define tile prefabs which fill a 1x1x1 unit of space so that they can be scaled to fit any user specified cell size (see "Use Cell Size" scale mode).

6. Choose appropriate option for **Tiles Facing**.

Tiles are rotated so that they face away from the tile system making it easier to fine tune rotation when when **Apply Prefab Transforms** is used.

Sideways

Recommended for side-scrolling maps.

Upwards

Recommended for top-down maps.

7. Review and adjust additional properties as needed.

8. Click **Create**.

Results

A new tile system should have been added to your scene.



Tip

Double-click the name of your new tile system using the scene palette or hierarchy window to locate it if out of view.

Tile System Creation Presets

Presets make it easy to reuse parameters that are commonly used when creating tile systems. Saved presets can be accessed from the **Preset** popup field.



Tip

Presets are stored in an XML file within your project directory `Assets/Rotorz/Tile System/UserData/preset.xml`. This file can be copied into other projects if needed.

Creating a Preset

The parameters used to create a tile system can be saved into a preset which can later be reused when creating a tile system.

Before you begin

Ensure that **Create Tile System** window is displayed (see [Creating a Tile System](#) on page 28).

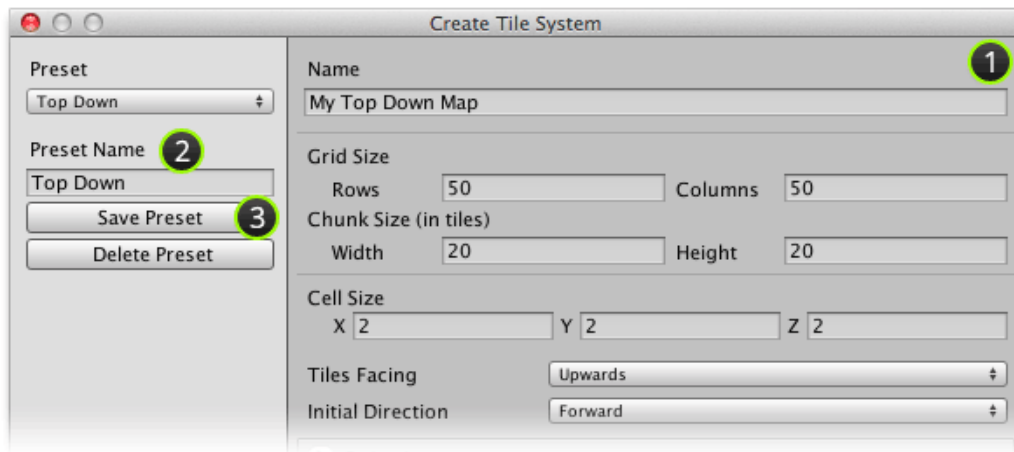


Figure 3-2. Creating a preset overview

Procedure

1. Input tile system creation parameters as required for preset.
2. Input name for preset.
3. Click **Save Preset** button to save preset.

Results

New preset should have been created and selected in **Preset** drop-down box.

Modifying a Preset

An existing preset can be modified when creating a tile system. Modifications can be then be saved for future re-use if desired.

Before you begin

Ensure that **Create Tile System** window is displayed (see [Creating a Tile System](#) on page 28).

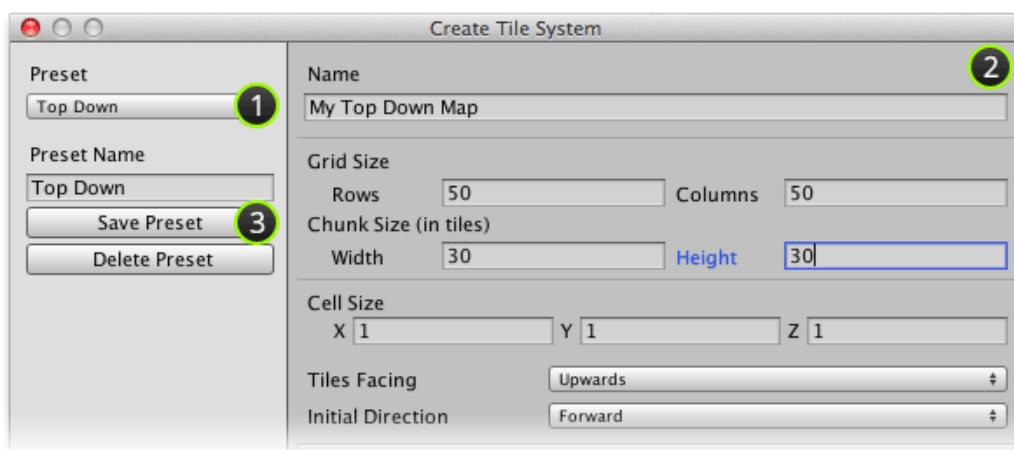


Figure 3-3. Modifying a preset overview

Procedure

1. Select the preset that you would like to modify using **Preset** drop-down box.
2. Make required alterations to tile system creation parameters.
3. Click **Save Preset** button to save changes to preset.

This step is optional if the modifications only apply to the tile system that you are in the process of creating.



Note

Changing a preset will **not** cause any changes to be made to previously created tile systems.

Results

Preset should have been updated with new parameters.

Deleting a Preset

An unwanted preset can be deleted using the **Create Tile System** window by selecting it and clicking the delete button.

Before you begin

Ensure that **Create Tile System** window is displayed (see [Creating a Tile System](#) on page 28).

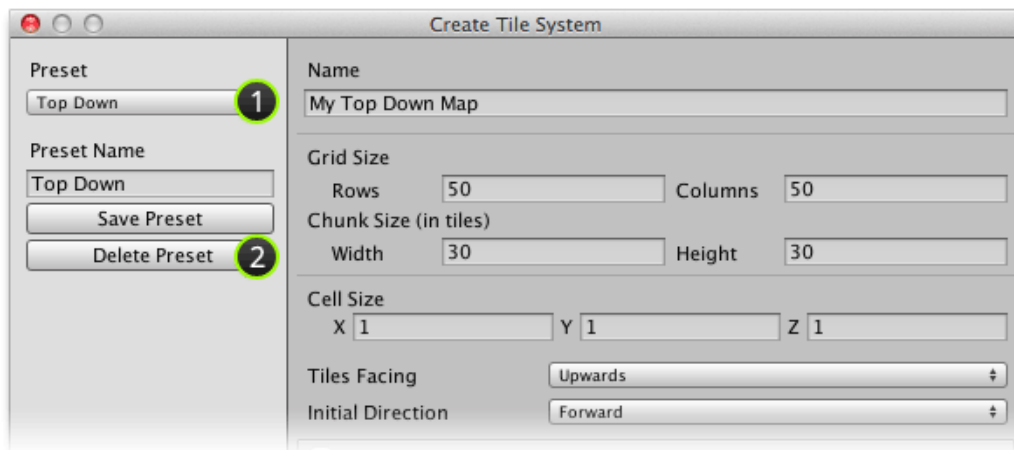


Figure 3-4. Deleting a preset overview

Procedure

1. Select the preset that you would like to delete using **Preset** drop-down box.
2. Click **Delete Preset** button.



Note

Deleting a preset will **not** cause any changes to be made to previously created tile systems.

Confirmation message should then be shown:

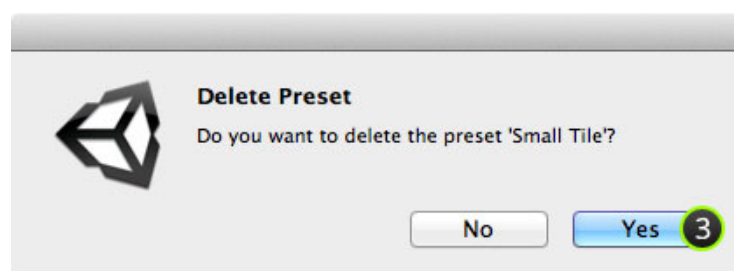


Figure 3-5. Delete preset confirmation message

3. Click **Yes** to delete preset.

Results

Preset should have been removed with the default preset selected.

Position, Rotate and Scale

A tile system can be selected using the scene palette or hierarchy window or by clicking its handle in scene view. Selected tile system can then be transformed using the regular move, rotate and scale tools provided by Unity.

Once tile system is positioned as desired you can paint on it as desired. It is often easier to paint using orthographic scene views; though this is entirely a matter of preference.

Figure 3-6. Screenshot of positioned, rotated and scaled tile system on page 32 demonstrates a tile system with all three transforms applied in edit mode. The small red wireframe cube highlights the active tile when painting.

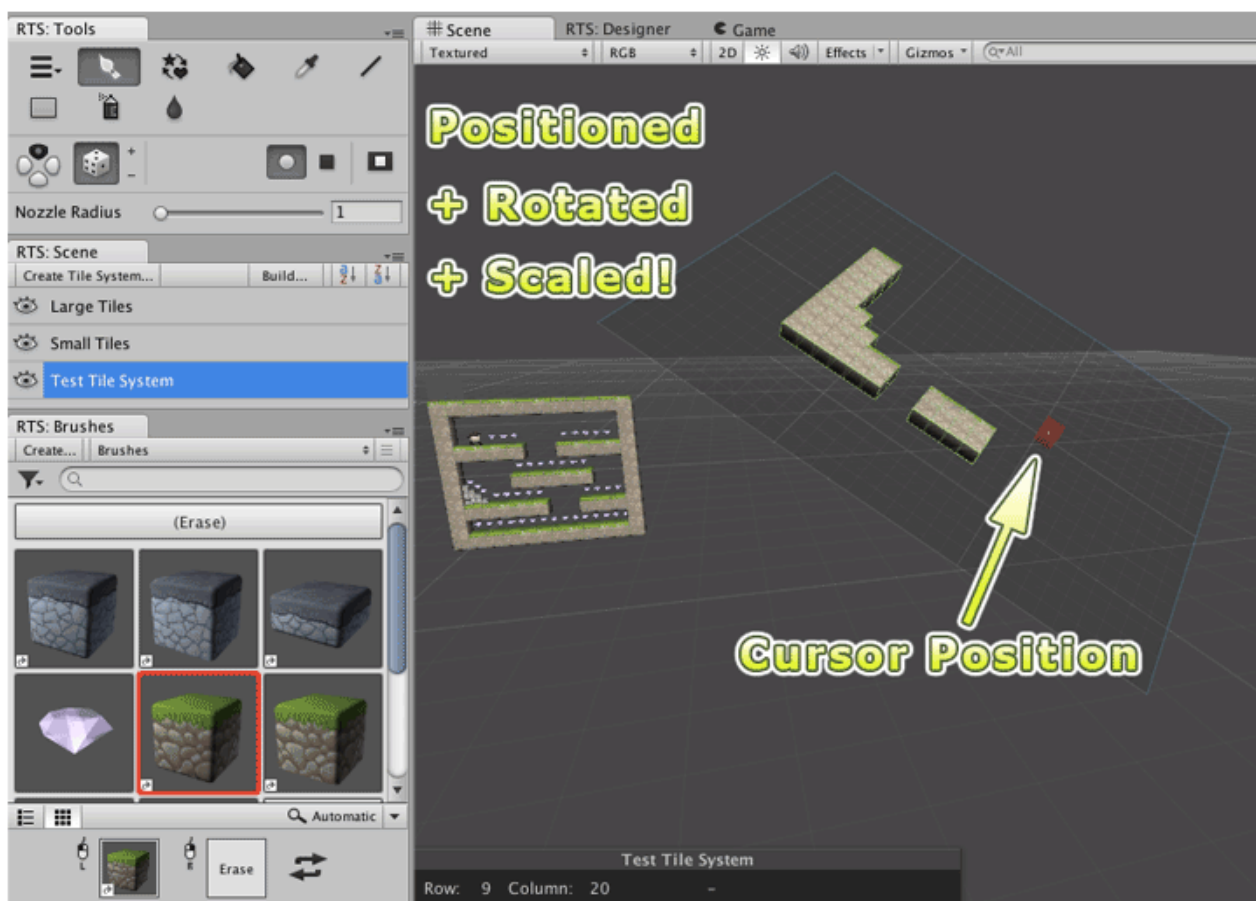


Figure 3-6. Screenshot of positioned, rotated and scaled tile system

Using the Inspector

A specialized inspector is provided allowing you to modify and configure tile systems after they have been created. Multiple tile systems can be inspected at the same time.



Accessing the Inspector

Procedure

1. Select the tile system that you would like to inspect.
2. Ensure that **Inspector** window is shown by selecting **Window | Inspector**.
3. Scroll to **Tile System** interface inside the **Inspector** window.

Inspector Interface

The inspector interface is split into sections which can be shown or hidden as needed. The more advanced functionality is hidden by default when Rotorz Tile System is first used.

The menu strip at the top of the interface allows you to save the tile system as a prefab, or alternatively save an optimized version as a prefab. Toggle visibility of grid lines by clicking , or visibility of chunk boundaries by clicking .

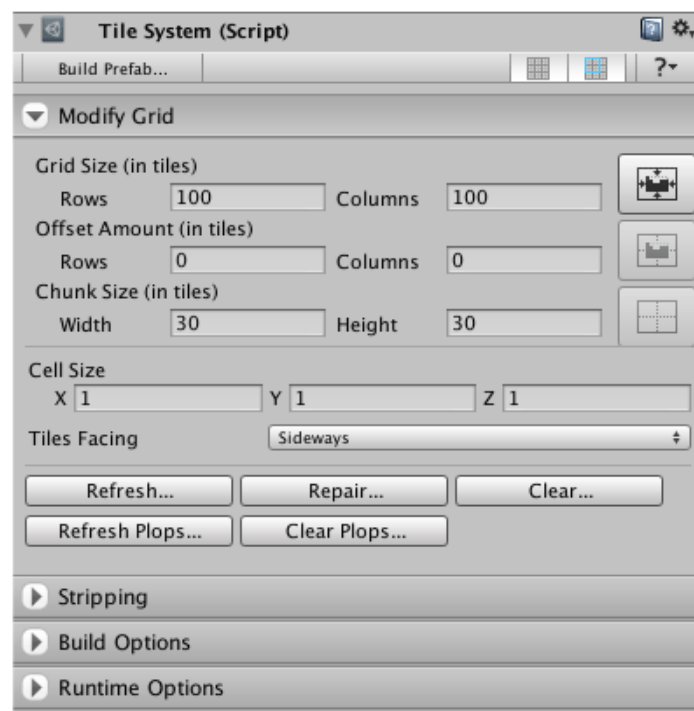


Figure 3-7. Inspector for a tile system

Changing Cell Size

You can change the width, height and depth of tile cells in an existing tile system using the inspector interface.

Before you begin



CAUTION

Some manual tweaks and changes may be lost including custom added components, or tile properties which have been modified using the inspector.

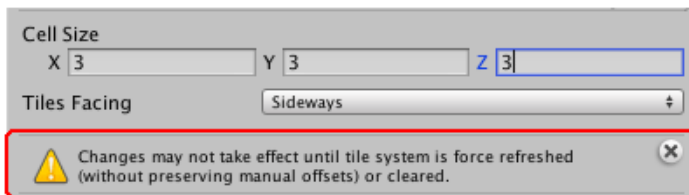
Ensure that **Inspector** window is shown by selecting **Window | Inspector**.

Procedure

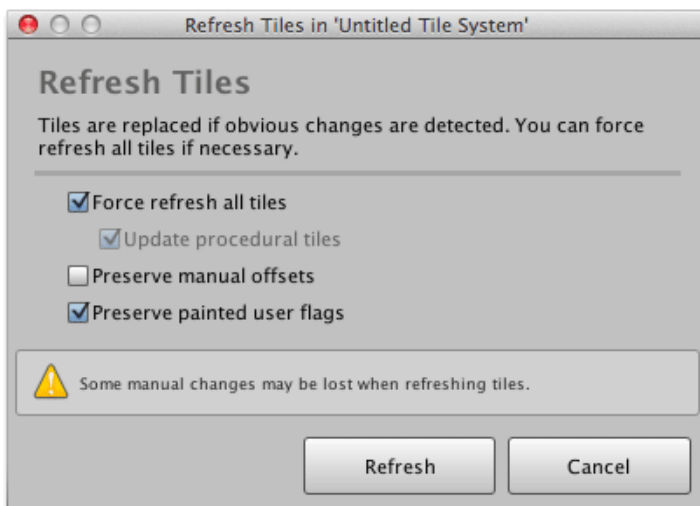
1. Select the tile system that you would like to change.
2. Input the new cell size using inspector.

You may notice that tile system grid is immediately updated in scene view, though existing tiles are not automatically updated.

A warning message will be shown indicating that some changes will not take effect until tile system is force refreshed or cleared. Tiles need to be force refreshed so that they are correctly positioned to fit within the updated grid:



3. Click **Refresh...** to refresh existing tiles.



- a. Ensure that both **Force refresh all tiles** and **Update procedural tiles** are selected to force tiles to be repainted.
- b. Deselect **Preserve manual offsets** so that position and scale of tiles are correctly updated.
- c. Click **Refresh**.

Results

Cell size has been updated and all previously painted tiles should have been updated to reflect this.

Repairing a Tile System

Tiles become broken when their data remains yet their game object counterpart is missing. A common cause of this is where tile game objects are deleted instead of being erased.

Before you begin

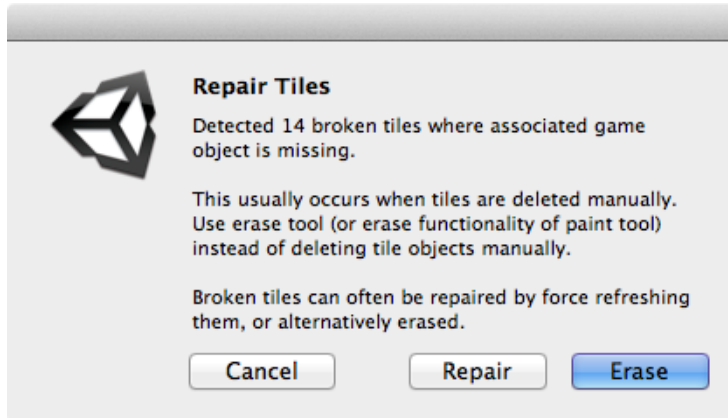
Ensure that **Inspector** window is shown by selecting **Window | Inspector**.

Procedure

1. Select your tile system.
2. Click **Repair...** button in tile system inspector.

There are two potential outcomes:

- No broken tiles were found, thus no further steps remain.
- Broken tile(s) found and a window similar to the following is displayed:



3. Choose from provided options:
 - **Repair** - Fix broken tiles by repainting with original brush (if still exists).
 - **Erase** - Erase broken tiles.
 - **Cancel** - Abort repair.

Clearing a Tile System

Tiles and chunks can be cleared from a tile system using the tile system inspector or via the context menu in the scene palette.

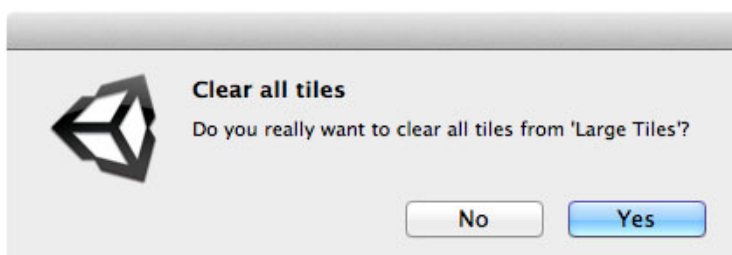
Before you begin

Ensure that **Inspector** window is shown by selecting **Window | Inspector**.

Procedure

1. Select your tile system.
2. Click **Clear...** button in tile system inspector.

A confirmation message should appear:



3. Select **Yes** to clear all tiles and chunks.

Optimization

Various options are provided allowing you to strip unwanted aspects from the tile systems in your projects. You can also combine tile meshes by material on a per chunk basis to reduce the number of draw calls which can be particularly beneficial when targeting mobile platforms.

Benefits of tile system optimization:

- Vertices of static tiles automatically snapped within user defined threshold.
- Smoothing of normals at tile joins.
- Removal of unwanted tile data and tile system components.
- Static tiles can be combined often resulting with fewer draw calls.



Note

The way in which a tile system is optimized can be configured using the **Inspector**.

Vertex Snapping

Often vertices around tile joins are supposed to occupy the exact same position but often don't due to small inaccuracies.

Inaccurate vertex alignment can lead to visual artifacts such as the glitter-like effect shown in [Figure 3-8](#). [Glitter-like effect caused by minor gaps between tiles on page 36](#). This undesirable effect can be eliminated by snapping the positions of vertices that are supposed to occupy the same space (provided that the design of the tiles allows for this).

Vertices within a given threshold (by default 0.001) will be snapped to occupy the same space. The threshold can be adjusted if necessary using the **Inspector** panel by selecting the tile system in question and altering the property **Vertex Snap Threshold**.

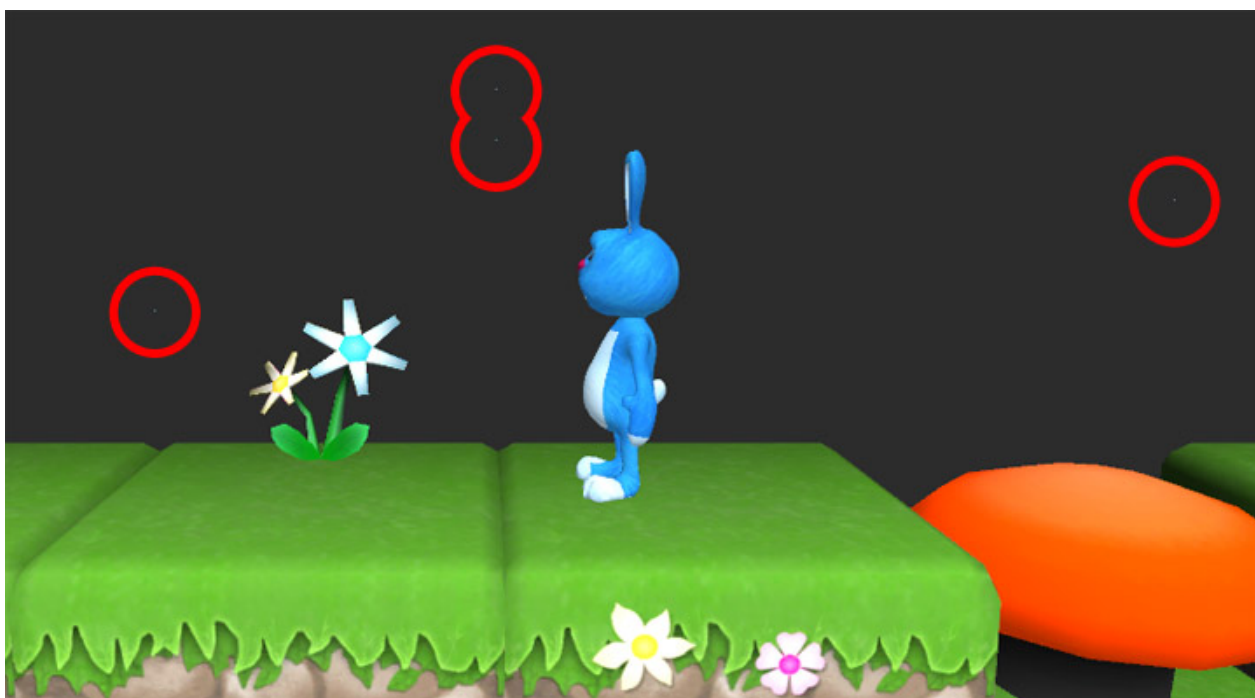


Figure 3-8. Glitter-like effect caused by minor gaps between tiles

**Note**

Background texture was removed from above illustration to make the glitter-like issue easier to see.

Smoothing

Whilst vertex snapping helps to avoid visual anomalies, you may still encounter issues regarding normals leading to lighting artifacts.

This issue is caused where the normals of adjacent tiles are not calculated smooth against each other. Modelling packages usually provide normal smoothing functionality, however it is not always possible to pre-smooth tiles that are designed to be used in a number of ways.

Figure 3-9. *Before and after tiles are smoothed* on page 37 shows how some tiles appear smooth whilst others do not. This can be corrected by marking the brushes as both **static** and **smooth**. Tiles painted using such brushes will be automatically smoothed using the vertex snapping threshold when tile system is built.

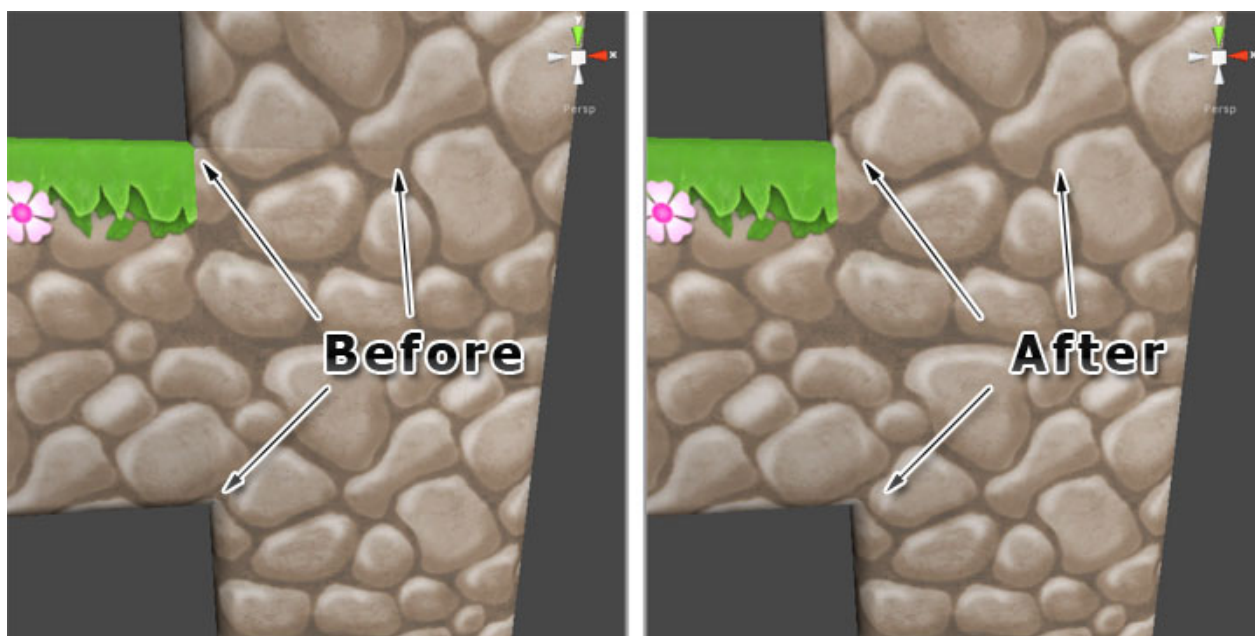


Figure 3-9. *Before and after tiles are smoothed*

Stripping

Tile systems contain a lot of data and functionality that is often not required at runtime (though is essential at design time). These unwanted aspects can be stripped upon building a tile system.



At runtime tile data is stored within tile system and chunk components so that both tiles and chunks can be accessed easily. This data can be discarded using the appropriate stripping options.

Whilst the majority of these stripping options are best appreciated for optimized tile systems, some of them can be applied at runtime when tile systems receive their first **Awake** message (see [Runtime Options](#) on page 41).

Stripping Presets

A number of stripping presets have been provided to cater for the most common use cases.

Stripping capabilities are reduced when procedurally generated tiles are present (unless they are pre-generated by selecting **Pre-generate Procedural** option). This is because tilesets and tile data are required to generate procedural tiles.

Stripping Preset	Description
Strip Runtime (default)	Remove runtime functionality.
Keep System Component	Remove runtime functionality but retain tile system component so that it is still possible to calculate the index of a tile from a position in world space, calculate tile index from ray casting, etc. <div>  Remember Tile data is unavailable! </div>
Runtime Access	Preserve ability to access tile data at runtime.
Runtime Painting	Preserve ability to paint tiles and access tile data at runtime.
Strip Everything	Perform maximum level of stripping.
No Stripping	Do not perform any stripping.
Custom	Define custom, see Stripping Options on page 38 . <div> <input checked="" type="checkbox"/> Strip tile system component <div> <input checked="" type="checkbox"/> Strip chunk map <input checked="" type="checkbox"/> Strip tile data <input checked="" type="checkbox"/> Strip brush references </div> <input type="checkbox"/> Strip empty objects <div> <input checked="" type="checkbox"/> Strip empty objects after combine </div> <input type="checkbox"/> Strip chunks and reparent tiles <div> <input checked="" type="checkbox"/> Strip empty chunks </div> </div> <p><i>Figure 3-10. Custom stripping options</i></p> <div>  Note Selecting or deselecting certain stripping options will automatically cause dependent options to be altered. For example, it is not possible to select "Strip tile system component" whilst at the same time deselecting "Strip chunk map". </div>

Stripping Options

Overview of custom stripping options that can be specified.



Stripping Option	Description
Strip tile system component	Removes <code>TileSystem</code> component from tile system game object.
Strip chunk map	Removes chunk map from tile system component and removes <code>Chunk</code> components from chunk game objects.
Strip tile data	Removes tile data from tile system chunk components.
Strip brush references	Clears brush references from tile data.

Stripping Option	Description
Strip empty objects	Removes most empty game objects.
Strip empty objects after combine	Removes game objects that have become empty after being combined with other tiles. Game objects will not be removed if they still contain components (like a collider, for example).
Strip chunks and reparent tiles	Removes chunk game objects and moves tile game objects into tile system game object.
Strip empty chunks	Removes chunk game objects that are empty or that have become empty during build process.
Strip plop instance / group components	<p>Special components are automatically added to plop instances and group objects allowing greater control over interaction with plops; for instance, allowing picking and erase.</p> <p>Whilst these components may be useful in custom scripts, they are not required at runtime and can be stripped to save memory.</p>

Build Options

Reference of options allowing you to customize the way in which your tile system is built.


Build Option	Description
Combine Method	<p>Runtime performance can often be improved by reducing the overall number of draw calls. The number of draw calls can be reduced by combining multiple tiles that share the same material.</p> <p>The performance of combined tiles can be further optimized by ensuring that a worthwhile amount of geometry is combined (whilst avoiding excessive numbers of vertices and triangles). This can be achieved by specifying appropriate chunk sizes when combining tiles. The chunk size will vary depending upon the complexity of your tiles along with your target platforms.</p> <p>None Do not attempt to combine tile meshes.</p> <p>By Chunk (default) Combine tiles into one mesh per chunk; with sub-mesh per material.</p> <p>By Tile System Combine tiles into one large chunk; with sub-mesh per material.</p> <p>Custom Chunk In Tiles Allows you to specify a different chunk size for tile combiner.</p>

Build Option	Description
Combine into Submeshes	<p>When tiles are merged with multiple materials you have the choice between combining them into submeshes or into separate meshes:</p> <p>Yes (default) - combine into submeshes (per chunk) A submesh is created for each material which according to the Unity documentation (Mesh.SetTriangles) are simply represented as separate triangle lists.</p> <p>No - combine into one mesh per material (per chunk) A separate mesh is created for each material which in some scenarios adds a little further mileage before the upper limit of 64k vertices is encountered.</p> <p>In both circumstances you will incur one draw call per material per chunk for simple single pass shaders.</p> <div>  Tip In the event that your combined chunks exceed the maximum limit of 64k vertices you can reduce the chunk size. Refer to Combine Method for further information. </div>
Vertex Snap Threshold	<p>Vertex snapping is performed on tiles that were painted using a smooth brush. The maximum snapping distance can be altered if needed. No vertex snapping or smoothing is performed on tiles when snapping threshold is set to zero.</p> <p><i>Default Value: 0.001</i></p>
Static Snapping	<p>Select this property when vertex snapping should also be performed on static tiles, or tiles that were painted using a static brush.</p>
Generate Second UVs	<p>Select this property when a second set of UV coordinates should be generated for static tiles. This can be useful when creating lightmaps.</p> <div>  Note Utilizes the built-in secondary UV generation functionality. See Unwrapping.GenerateSecondaryUVSet. </div>
Pre-generate Procedural	<p>Causes all tiles that are painted using procedural brushes to be pre-generated. Selecting this option allows tile data and brush references to be stripped. Procedurally generated meshes will be saved into the optimized scene which will naturally increase its file size.</p> <p>Procedural tiles will be generated at runtime if this option is not specified.</p>

Build Option	Description
Reduce Box Colliders	<p>Searches for adjacent box colliders which were painted using a static brush and then attempts to combine them to reduce the overall number of colliders. Box colliders can only be combined when their corner points connect.</p> <p>Snap Threshold Threshold which must be satisfied to determine whether two corner points occupy the same space.</p> <p>Keep Separate Colliders can be kept separate by tag and/or layer allowing collision detection logic to differentiate between colliders by tag and/or layer.</p> <p>Set to Nothing to reduce colliders regardless of tag and layer. This can lead to fewer colliders but may affect runtime behavior.</p> <p>Include tiles flagged as solid Tiles which are flagged as "solid" can be promoted into box colliders allowing them to be reduced with actual colliders. This helps to reduce the number of colliders in your scene since colliders would otherwise form around a gap.</p> <p>The "Grass Platform" demonstration brush is an example of a brush which benefits from this.</p>

Runtime Options

Reference of options that apply to runtime.

Runtime Option	Description
Erase Empty Chunks	<p>When ticked, hints that empty chunks should be erased when they become empty when erasing tiles.</p> <div>  <p>Note The value of this option is also respected when editing tile system at design time.</p> </div>
Apply Basic Stripping	Indicates whether a basic level of stripping should be applied when tile system receives the Awake message.
Update Procedural at Start	When ticked, procedural tiles will be updated when the tile system receives its first Awake message.

Runtime Option	Description
Mark Procedural Dynamic	<p>Indicates whether procedurally generated meshes are likely to be updated throughout play where tiles are painted and/or erased frequently to improve performance (this may use more memory).</p> <p>It is useful to mark procedural meshes as dynamic when painting or erasing tiles as part of game-play or a custom in-game level designer.</p> <p>Avoid setting if procedural meshes are only updated at start of level when loading or generating map.</p>
Add Procedural Normals	When ticked, indicates that normals should be added to procedural meshes which allows you to use shaders that require normals. This is required if you would like procedural tiles to respond to lighting.
Procedural Sorting Layer	The sorting layer that should be used for procedural tileset meshes.
Procedural Order in Layer	Order in sorting layer to use for procedural tileset meshes.

Building Prefab from Tile System

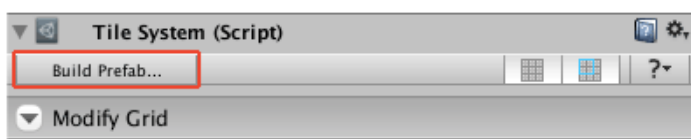
Sometimes it is useful to save optimized versions of tile systems as prefabs which can then be composed to make scenes. Additional mesh assets will be generated when tiles are combined.

Before you begin

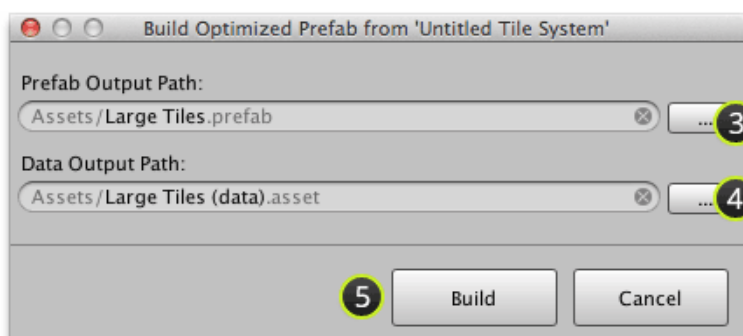
Ensure that **Inspector** window is shown by selecting **Window | Inspector**.

Procedure

1. Select your tile system.
2. Click **Build Prefab...** command at start of **Tile System** interface:



The following window should then appear:



3. Click first ... button to specify output path for generated prefab.

**Note**

This will automatically pre-fill the data output path. You may skip the following step if you are happy with the automatically filled path.

4. Click second ... button to specify output path for generated data asset.
5. Click **Build**.

Results

Optimized version of tile system should have been saved to prefab with accompanying data asset whilst the original tile system is left untouched.

Building Entire Scene

Rotorz Tile System provides an option allowing you to build all of the tile systems in the current scene which will then be saved to a separate scene file. Original scene should be retained so that you can make changes and then rebuild.

Before you begin

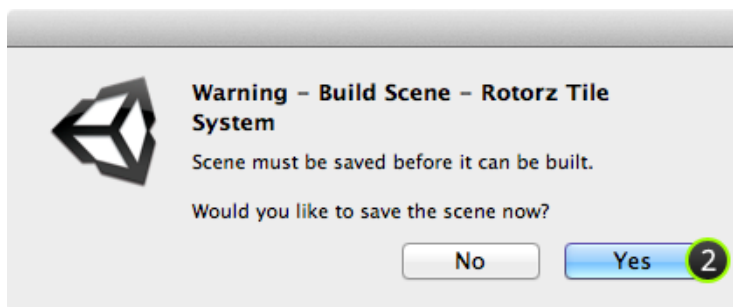
**CAUTION**

Current scene must be saved before it can be built. Also note that previous build will be lost when overwriting an existing scene file.

Procedure

1. Select menu  | **Build Scene...**

The following window should appear:

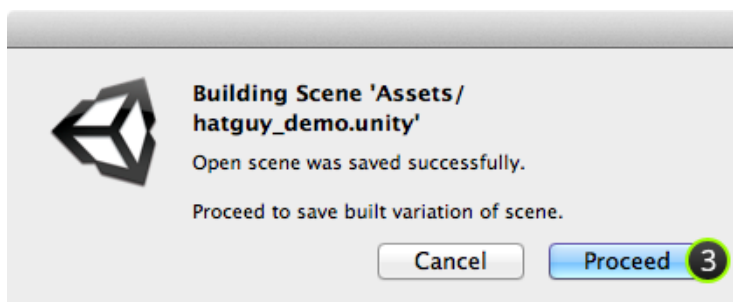


2. Select **Yes** to save current scene before attempting to build optimized version.

**CAUTION**

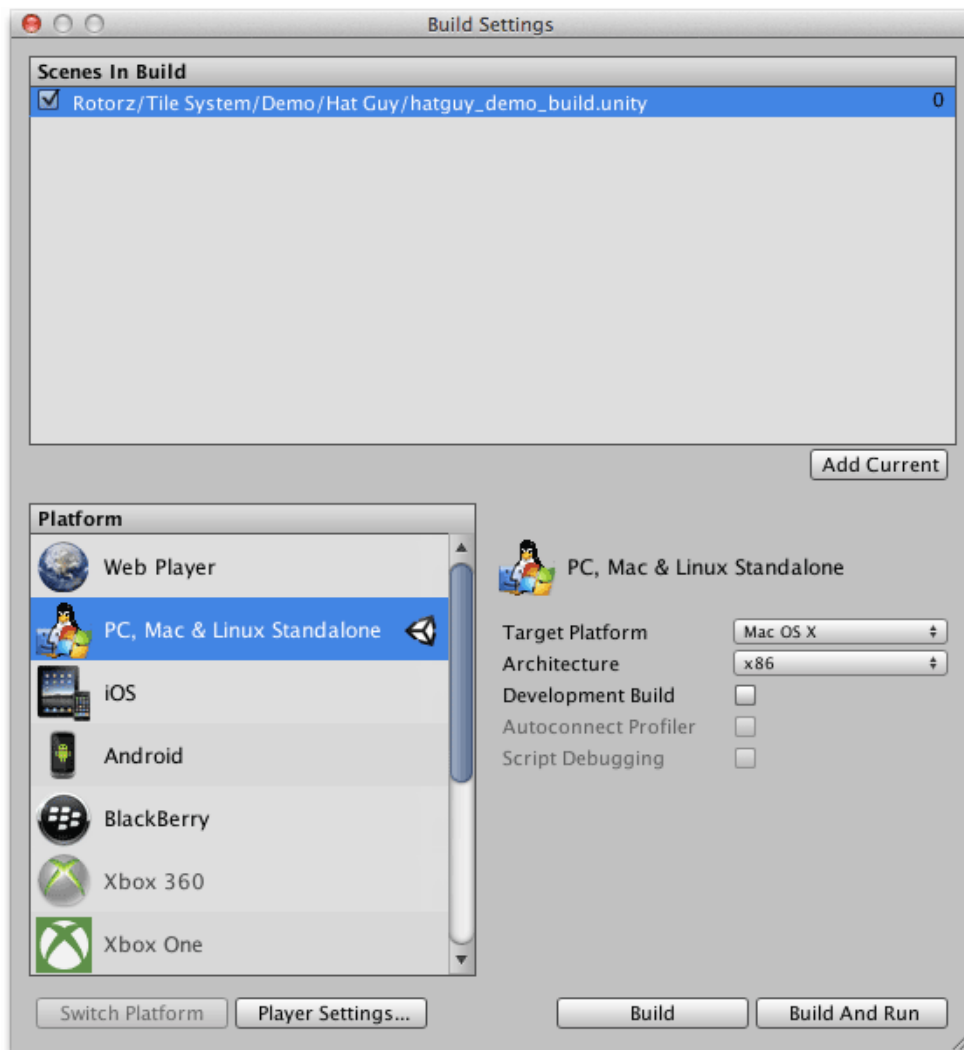
Undo/redo history will be lost during build process. Do not proceed if you do not wish to save your scene.

You should then be prompted with the following message:



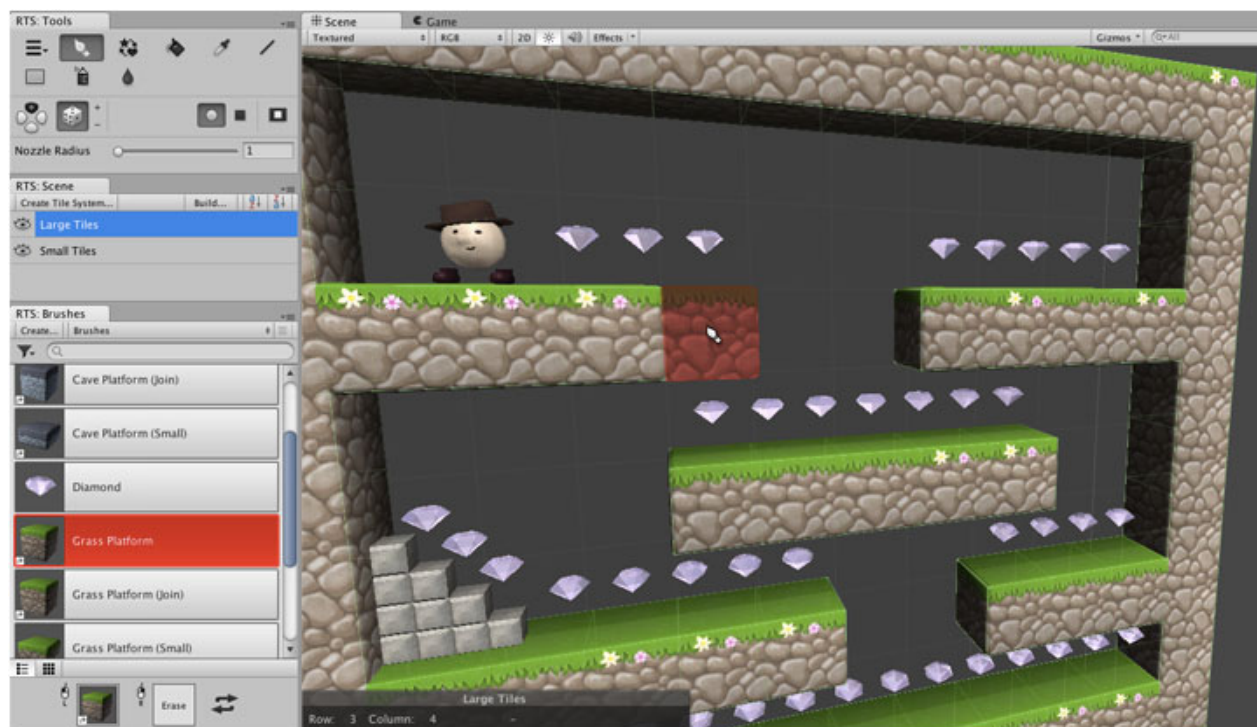
3. Select **Proceed**.
4. Specify output file for optimized version of scene.
5. (Optional) Add optimized version of scene to project build.
 - a. Open optimized scene using **Project** window.
 - b. Select menu **File | Build Settings**.
 - c. Click **Add Current** button.

You should then see the path of your optimized scene listed:



Chapter 4.

Painting



Tools

Rotorz Tile System provides a number of tools allowing you to paint tiles from within the Unity editor. Some tools have additional parameters which are shown in the tool palette. Tools can be accessed via the tool palette **Window | Rotorz Tile System**.

Some tools can be quickly activated when a tile system is selected using a keyboard shortcut. Refer to tool topics to find out which tools have shortcuts.



Fastpath

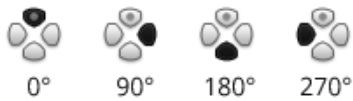
Quickly select paint tool by holding **Ctrl** whilst **Left Clicking** brush.

Common Tool Options

Some options are common to a number of tools.

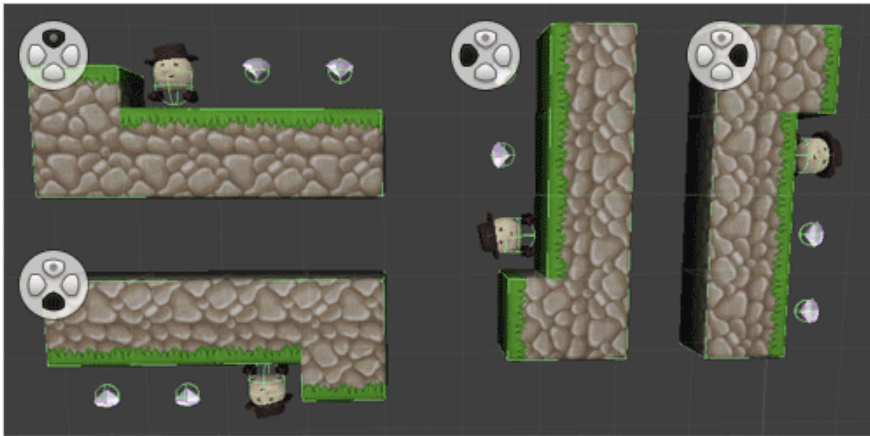
Rotation Selector

Paint tiles with simple rotation at 90 degree intervals. This feature works with 2D and 3D tiles. Simply click on the button representing the desired rotation to select it.

**Shortcut Keys:**

- **M** - Reset to 0 degrees
- **,** (comma) - Rotate anti-clockwise by 90 degrees
- **.** (period) - Rotate clockwise by 90 degrees

The following image demonstrates tiles which have been painted with various rotation selections:

**Tip**

Behavior of rotation is affected by the default rotation of a tile. In the case of 3D tiles this can be adjusted by setting the prefab offset.

**Randomize Variation**

Oriented brushes allow you to specify multiple variations for each orientation which can be picked at random when painting.

+

- Shift to Next or Previous Variation

Sometimes it is useful to shift to the next or previous variation before painting or plopping a tile (i.e. cycle before painting).

Shortcut Keys:

- **+** - Shift to next variation
- **-** - Shift to previous variation
- **0** - Reset shift to zero
- **Alt + Mouse Wheel** - Shift to next or previous variation

**Paint Around Existing Tiles**

Tiles will be painted around existing tiles when this option is selected. This option does not apply when erasing tiles.

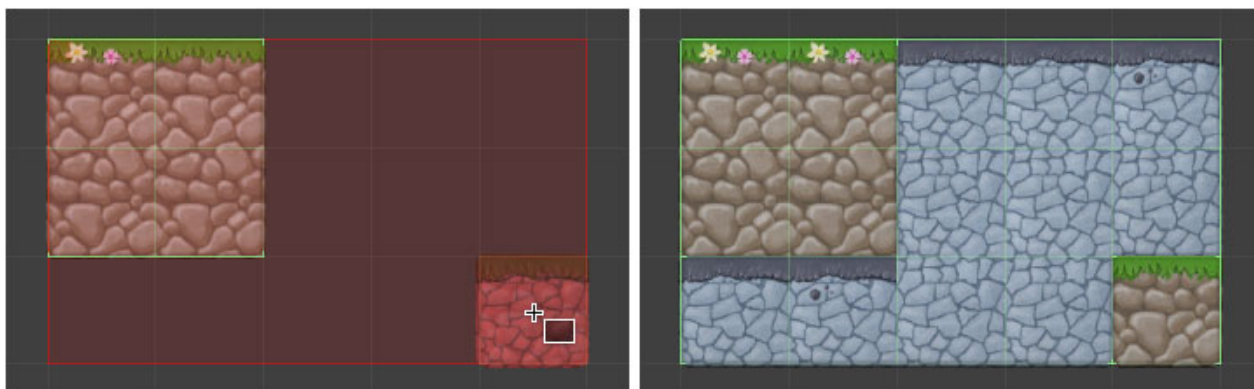


Figure 4-1. Painting filled rectangle around existing tiles.

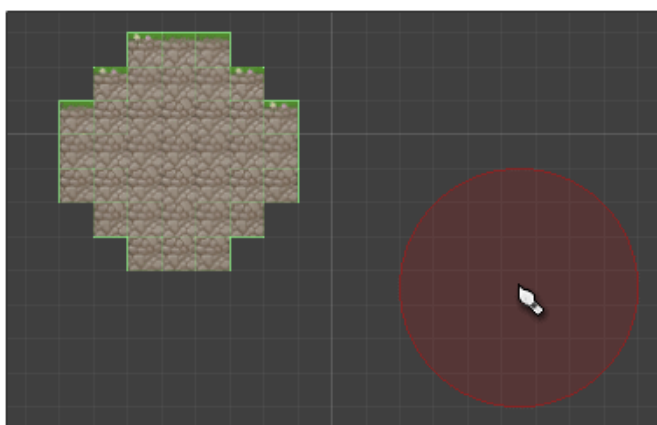
Nozzle Shape

Shape of nozzle to use when painting tiles:

- ● Round Nozzle
- ■ Square Nozzle

Nozzle Radius / Nozzle Size

Radius or size of nozzle can be adjusted using a slider depending upon which nozzle shape is selected. Nozzle area is highlighted when painting like illustrated below:



Shortcut Keys:

- [- Decrease radius or size of nozzle
-] - Increase radius or size of nozzle
- **Ctrl + Mouse Wheel** - Increase or decrease radius or size of nozzle

Deactivate Tool

Select one of the Unity provided tools to deactivate tile system tool or re-click active tool button to restore previously selected Unity tool.

Or use one of the following keyboard shortcuts:

- **Q** - Hand Tool
- **W** - Move Tool
- **E** - Rotate Tool

- **R** - Scale Tool

Paint Tool

Single click or drag to paint tiles onto active tile system.

- Use **left mouse** button to paint using primary brush.
- Use **right mouse** button to paint using secondary brush.

Shortcut Key: With tile system selected press **B** to select tool.



Tip

Left click on painted tile to cycle through to next available variation when brush radius is set to 1.

Paint Line of Tiles

Hold **Shift** key to paint line of tiles from previously painted tile to cursor. Visual representation of line is shown whilst key is held as illustrated below:

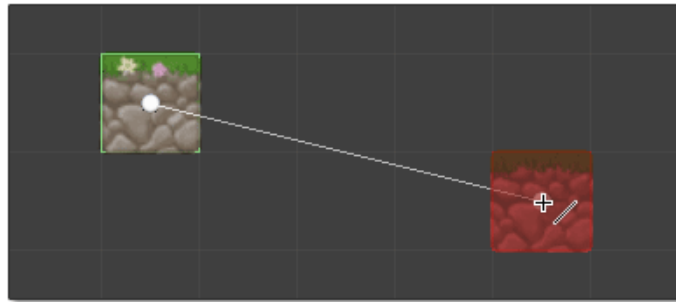


Figure 4-2. Paint straight line by holding shift key.

Paint Straight Line of Tiles

Hold both **Shift** and **Ctrl** keys to paint horizontal or vertical line of tiles more easily:

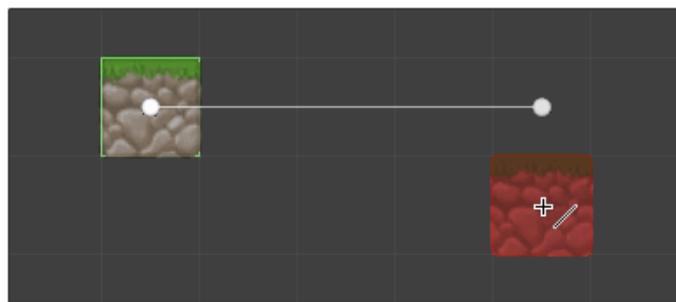


Figure 4-3. Paint horizontal line by holding both shift and control keys.

Cycle Tool

Cycle through available variations of a tile or plop that has already been painted.

Oriented brushes allow you to provide multiple variations which can be randomly chosen whilst painting. This tool allows you to manually cycle through available variations allowing you to pick the one that you want.

- **Left Click** - Cycle to previous variation.
- **Right Click** - Cycle to next variation.

Advanced Options

Can cycle plops

Indicates whether plops can be cycled when using cycle tool. Deselect this option when plops are stealing input focus from tile system making it difficult to cycle regular tiles.

Interact with active system only

Indicates whether cycle tool should only be able to interact with plops that are associated with the active tile system.

Fill Tool

Flood fill area of active tile system.

- **Left Click** - Flood fill using primary brush.
- **Right Click** - Flood fill using secondary brush.

Shortcut Key: With tile system selected press **G** to select tool.

Advanced Options

Fill Limit

Indicates the maximum number of tiles which should be painted when using fill tool (set to 300 by default). Filling too many tiles may cause Unity to crash; so avoid setting this option too high!

Picker Tool

Pick primary or secondary brush from a previously painted tile and then automatically switch to the paint tool.

- **Left Click** - Pick primary brush selection from painted tile.
- **Right Click** - Pick secondary brush selection from painted tile.

Shortcut Key: With tile system selected press **I** to select tool.

Advanced Options

Can pick plops

Indicates whether brush can be picked from plops when using picker tool. Deselect this option when plops are stealing input focus from tile system making it difficult to pick from regular tiles.

Interact with active system only

Indicates whether picker tool should only be able to interact with plops that are associated with the active tile system.

Line Tool

Paints a straight line of tiles by dragging with left or right mouse button pressed.

- Use **left mouse** button to paint tiles using primary brush.
- Use **right mouse** button to paint tiles using secondary brush.

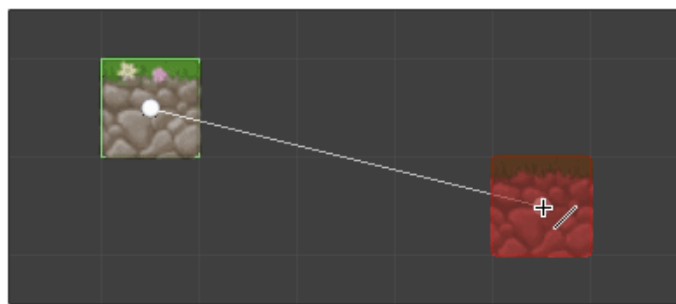


Figure 4-4. Painting a line using the line tool.

Hold **Shift** key to draw a straight line horizontally or vertically.

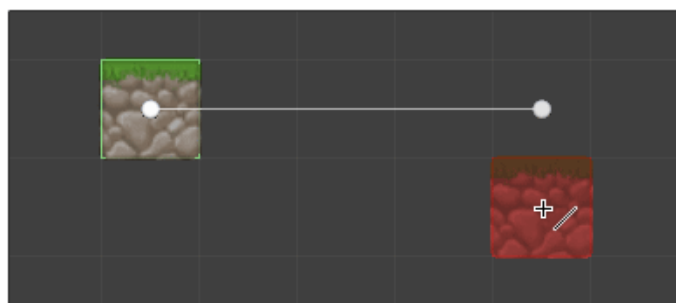


Figure 4-5. Painting a horizontal line by holding the shift key using the line tool.

Rectangle Tool

Paints filled or outline of rectangle by dragging with left or right mouse button pressed.

- Use **left mouse** button to paint tiles using primary brush.
- Use **right mouse** button to paint tiles using secondary brush.

Shortcut Key: With tile system selected press **U** to select tool.

Fill Mode

-  Center area of rectangle is filled when selected:

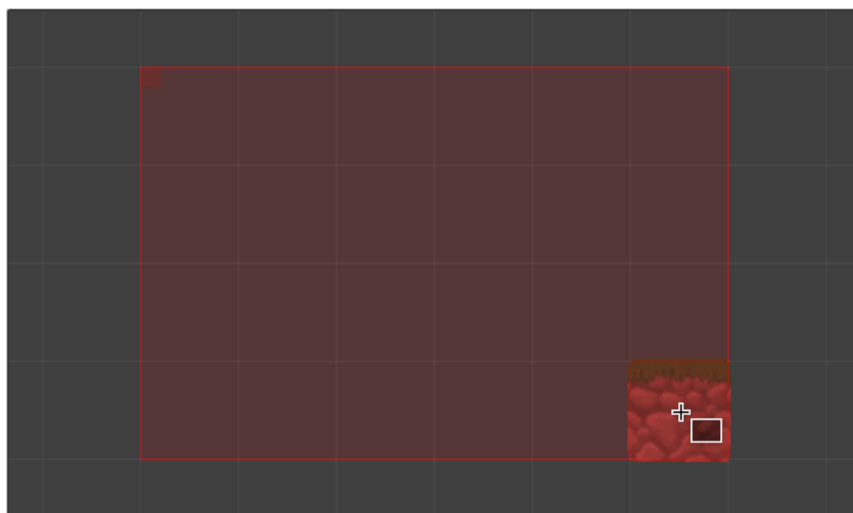


Figure 4-6. Painting filled rectangle.

-  Outline of rectangle is drawn when selected:

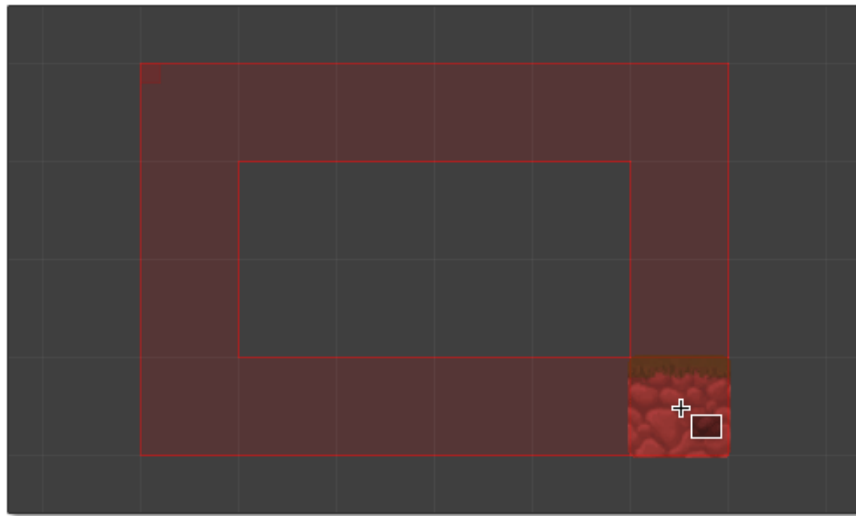


Figure 4-7. Painting outline of rectangle.

Spray Tool

Use spray tool to randomly scatter tiles onto a tile system.

- Use **left mouse** button to paint tiles using primary brush.
- Use **right mouse** button to paint tiles using secondary brush.

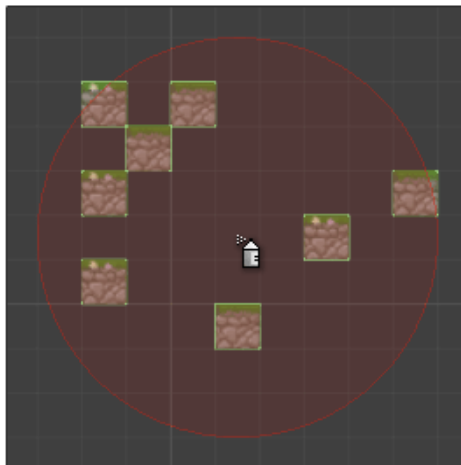


Figure 4-8. Spray some tiles with round nozzle.

Fill Rate (%)

Rate at which to fill nozzle area with tiles. Use lower value to reduce quantity of tiles painted.

Paint Line of Tiles

Hold **Shift** key to paint line of tiles from previously painted tile to cursor. Visual representation of line is shown whilst key is held as illustrated below:

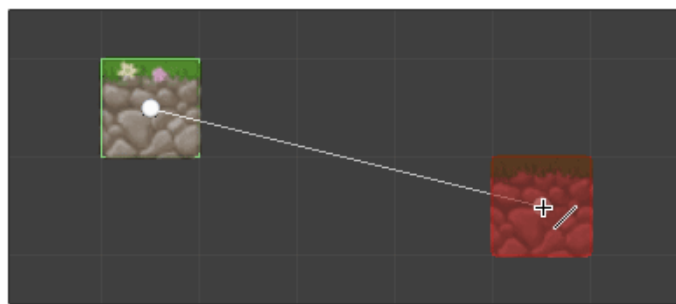


Figure 4-9. Paint straight line by holding shift key.

Paint Straight Line of Tiles

Hold both **Shift** and **Ctrl** keys to paint horizontal or vertical line of tiles more easily:

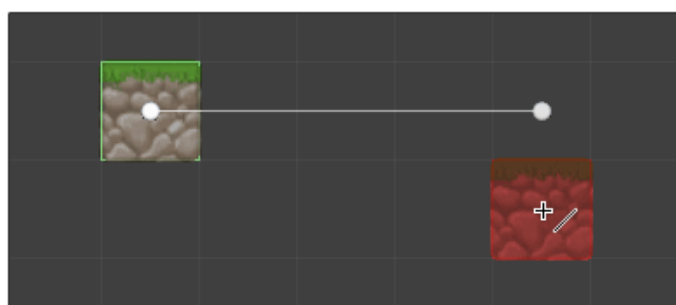


Figure 4-10. Paint horizontal line by holding both shift and control keys.

💧 Plop Tool

Plops object somewhere on plane of tile system using brush but does not assign to data structure of tile system.

- Use **left mouse** button to plop tiles using primary brush.
- Use **right mouse** button to plop tiles using secondary brush.

Shortcut Key: With tile system selected press **P** to select tool.



Tip

Left click on painted plop to cycle through to next available variation.

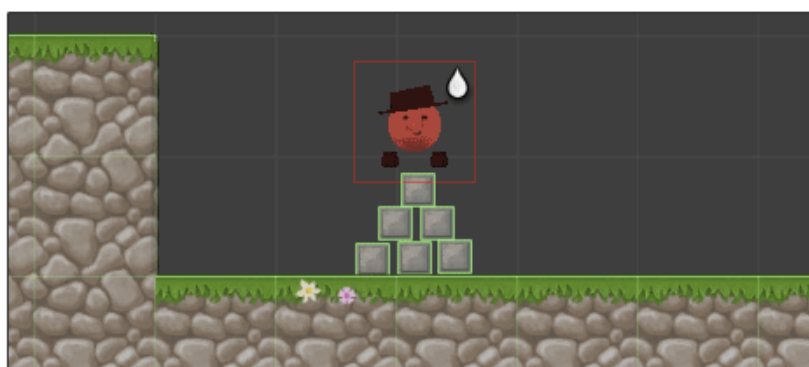


Figure 4-11. Freely plopping tiles onto a tile system.

Alignment on X and Y Axis

Plops can be freely placed or aligned to a subdivision of the active tile system's grid. Separate alignments can be specified for the X and Y axis though these are linked by default (∞).

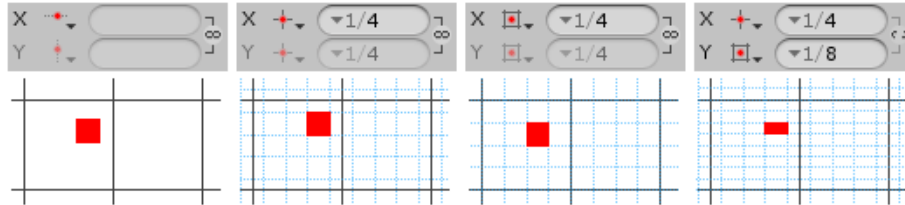


Figure 4-12. Plop alignment examples.

Snap Alignment

- **Free** - Free placement of plops.
- **Points** - Snap plops to points (where grid lines intersect).
- **Cells** - Snap to midpoint of grid cells.

Snap Grid Spacing

When snapping to grid points or cells the spacing between grid cells can be specified using one of the following modes which can be selected using the drop-down arrow to left of input field.

- **Fraction of Cell Size** - Make snapping grid relative to cell size of the active tile system.
- **Custom** - Specify custom grid in local space of the active tile system.

Common presets can also be quickly selected from the drop-down.

Plopping Overlapping Objects

Hold **Ctrl** key to temporarily disable cycle functionality of plop tool so that you can plop objects on top of one another.



Tip

You may want to consider adjusting the advanced option **Disable cycle function** if you rarely use cycle functionality of plop tool.

Advanced Options

Location

Location for new plops:

- **Group Inside Tile System** - Group plops into game object inside tile system.
- **Child Of Tile System** - Place plops inside tile system game object.
- **Scene Root** - Place plops at scene root.

Disable cycle function

Disables cycle functionality of plop making it possible to plop overlapping tiles without needing to hold **Ctrl** key.

Interact with active system only

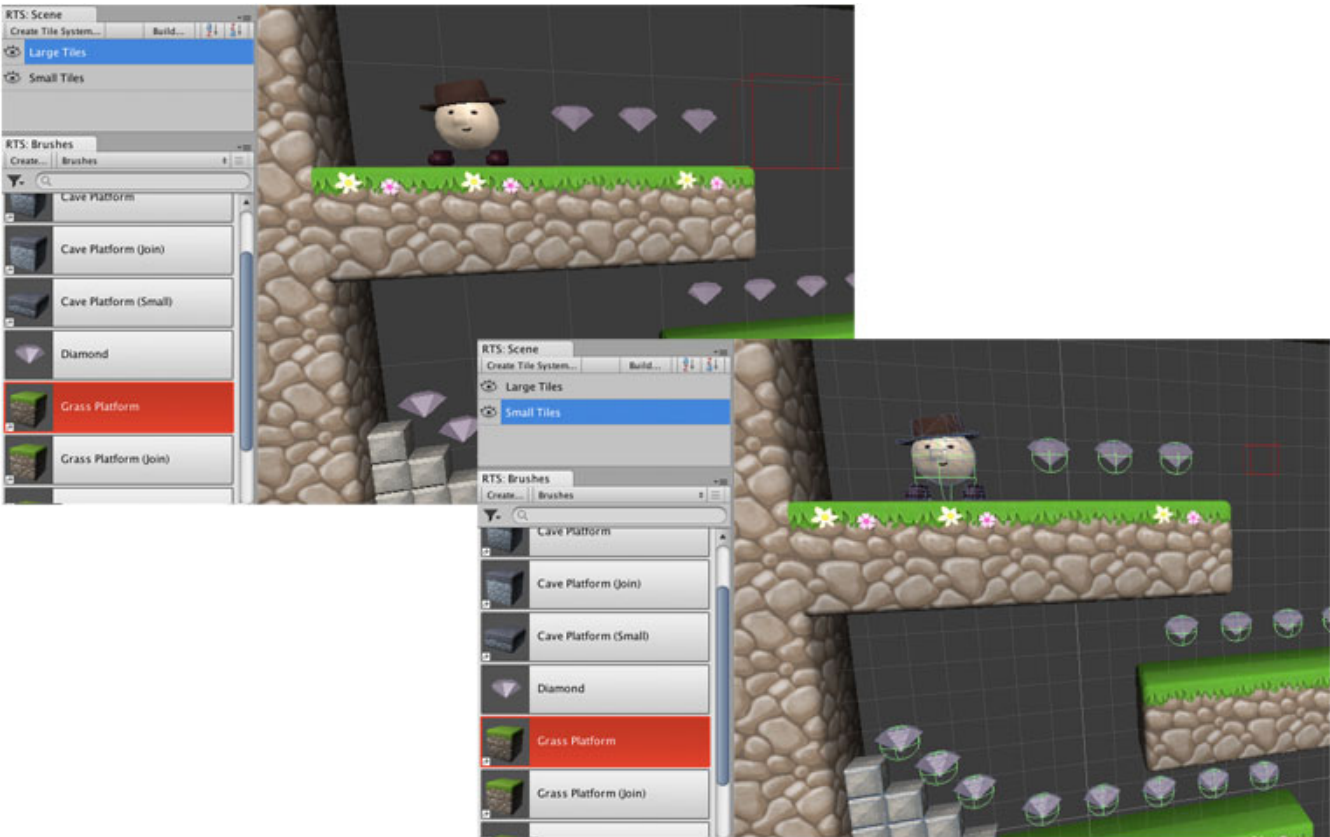
Indicates whether erase and cycle functionality should only be able to interact with plops that are associated with the active tile system.


Hide wireframe outline

Indicates whether wireframe outline should be shown to highlight position of plop. Wireframe outline is always shown for brushes where immediate previews have been disabled.

Switching Between Tile Systems

When your scene contains a number of tile systems it is useful to be able to switch between them quickly when painting.



The [Scene Palette on page 8](#) ( | **Editor Windows** | **Scene**) makes it easier to switch between tile systems when painting. More importantly it remembers the previous tile system that you painted on allowing you to quickly resume painting simply by selecting a tool.

Keyboard Shortcuts

The following keyboard shortcuts can be used within scene views when a tool is selected.

Shortcut	Description
Page Up	Select previous tile system using scene palette order.
Page Down	Select next tile system using scene palette order.

You can also assign keyboard shortcuts to make it even easier to switch to and from specific tile systems. Please refer to the Unity documentation for further details: <http://docs.unity3d.com/Manual/UnityHotkeys.html>.

Status Panel

The status panel reflects the state of the highlighted tile when a tool is selected. This is useful when developing oriented brushes.

The status panel reflects the state of the active tile when a tool is selected. The active tile is highlighted with a red wireframe box near the mouse pointer when looking at the scene view. The status panel is broken down as shown in [Figure 4-13. Annotated illustration of status panel on page 56.](#)

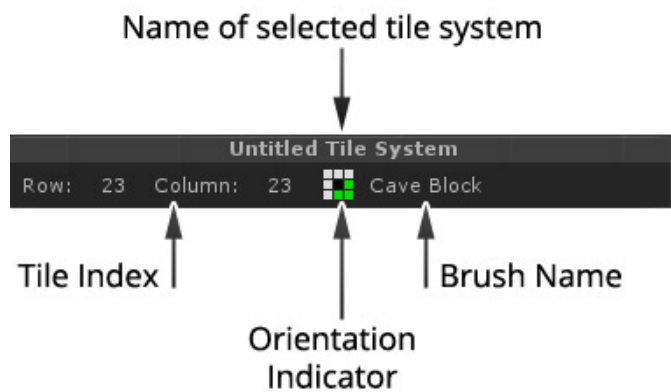



Figure 4-13. Annotated illustration of status panel

Orientation Indicators

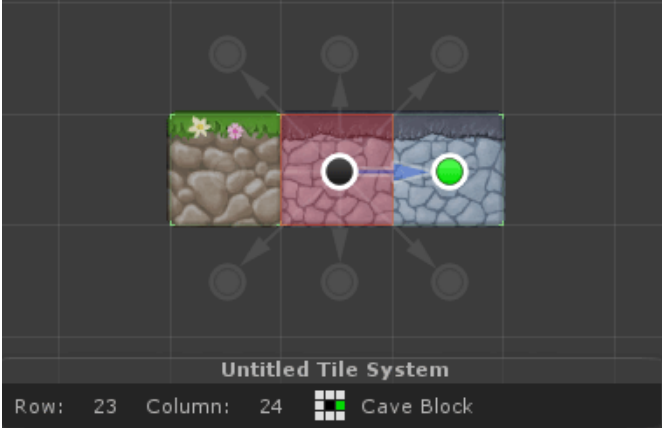
The orientation indicator is often useful when developing oriented brushes because it shows which brush orientation was used along with the actual orientation of the tile. By looking at the colors presented it is possible to determine:

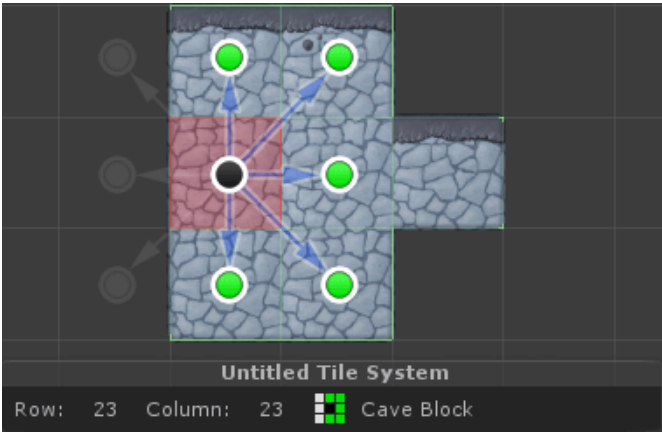
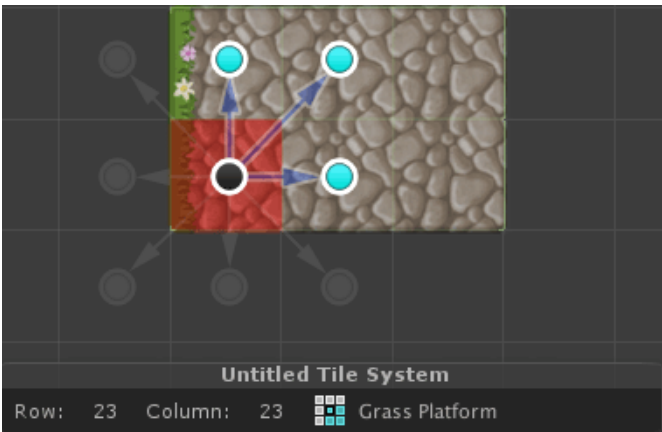
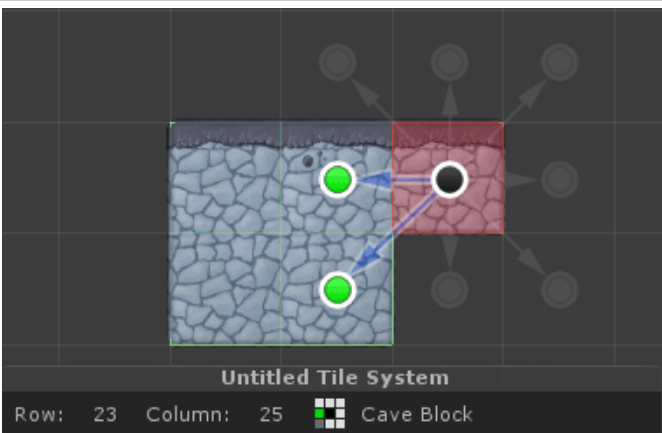
- When a fallback orientation was used.
- Which fallback orientation was used.
- The actual orientation of the tile.

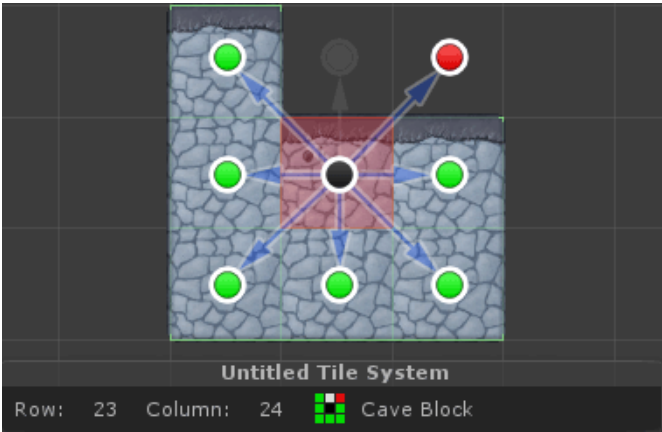
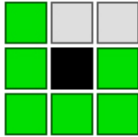
**Remember**

Whilst the presence of red and gray indicators hint that brush does not contain the actual orientation, it does not necessarily mean that there is an error. Though this can be extremely useful when you see that the wrong tile has been painted because you can see what went wrong.

The following legend describes what each of the colors indicate:

Orientation Indicator	Meaning
	White Indicates that the surrounding tile is either empty or that the coalescing rules of the tile exclude it from the orientation of the active tile. For example, in the illustration to the left you can see that the active tile coalesces with the tile to its right because it is another "Cave" tile. However the block to its left is a "Grass" tile that is configured to coalesce with other "Grass" tiles only.

Orientation Indicator	Meaning
	<p>Green</p> <p>Indicates that the surrounding tile coalesces with the active tile, and that the brush orientation that was used to paint the active tile matches the surrounding tile.</p> <p>If indicator is entirely white and green then the painted tile was matched exactly as defined within brush.</p>
	<p>Cyan</p> <p>Same as with green except indicates that tile was painted with rotation.</p>
	<p>Gray</p> <p>Indicates that the surrounding tile is a part of the actual orientation of the active tile, but the brush orientation that was used to paint the active tile did not match the surrounding tile.</p> <p>If the active tile was not appropriate in this example, it would be necessary to define the following orientation:</p> <div data-bbox="855 1514 992 1648"></div>

Orientation Indicator	Meaning
	<p>Red</p> <p>Indicates that the surrounding tile is not part of the actual orientation of the active tile, but the nearest available brush orientation that was used to paint the active tile did match the surrounding tile.</p> <p>If the active tile was not appropriate in this example, it would be necessary to define the following orientation:</p> 

Tweaking Tiles

Non-procedural tiles can be manually tweaked, moved, rotated and scaled once they have been painted.

It is also possible to add, remove and tweak the properties of painted tile components using the inspector. This is sometimes useful when painting various active items such as collectables, enemies, trigger points, etc.

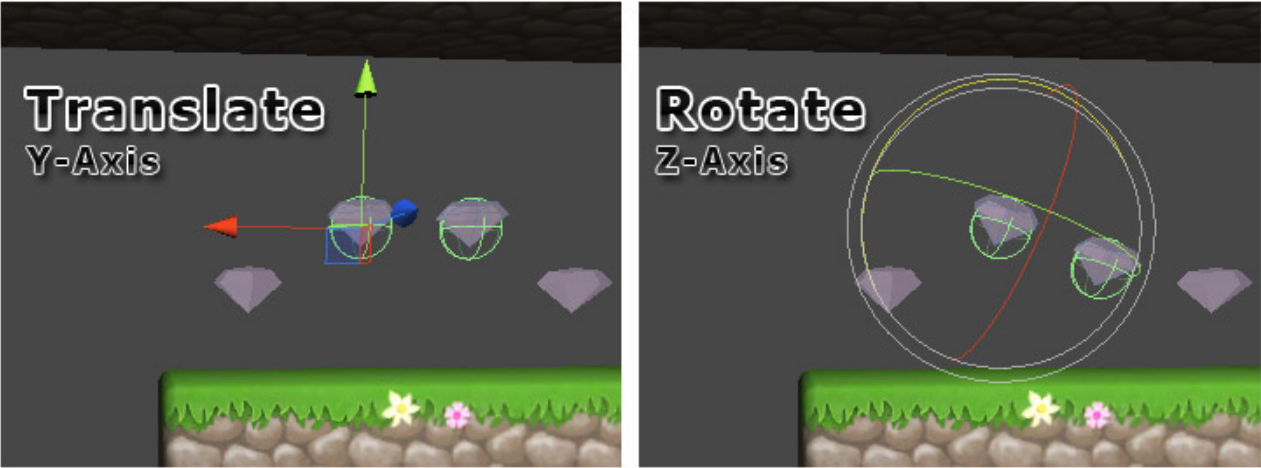



Figure 4-14. Example of tiles that have been manually tweaked

**Note**

Manual tweaks will usually be retained, however some actions may cause such changes to be lost.

Replace by Brush

You can quickly repaint tiles which were painted using one brush with another brush. Think of this as find and replace for tiles!

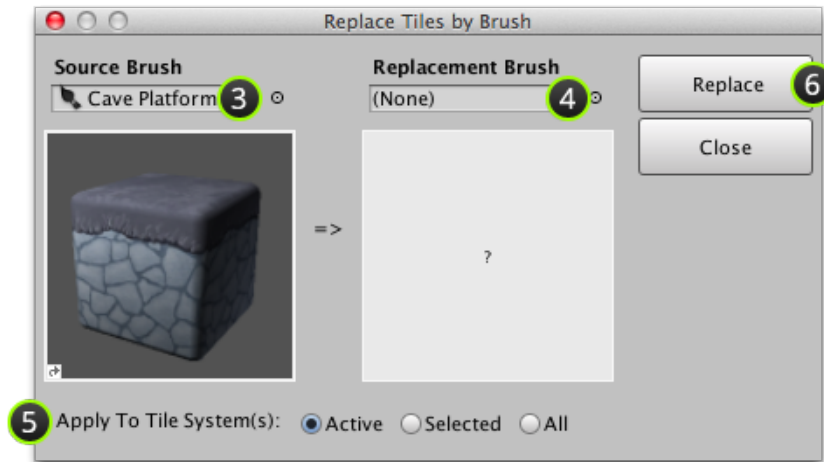
Procedure

1. (Optional) Select tile system(s) that you would like to alter.

Skip this step if you would like to find and replace tiles for all tile systems in scene.

2. Select menu command  | **Replace by Brush....**

The following window should then be shown:



3. (Optional) Select **Source Brush**.

Field is automatically populated with primary brush.

4. Select **Replacement Brush**.

All instances of source brush will be repainted with this brush.

5. Specify tile system(s) which find and replace should operate on:

- **Active** - The active tile system.
- **Selected** - All editable tile systems in current selection.
- **All** - All editable tile systems in scene.



CAUTION

Always verify that this option is set correctly.

6. Click **Replace** to perform find and replace.

Refreshing a Single Tile

An existing tile may appear stale when changes have been made to its associated prefab or the brush that was used to paint it. A tile can be updated by repainting it.

Procedure

1. Erase tile using **Erase** brush or by right-clicking when using the **Paint** tool.
2. Repaint tile using the desired brush.

Results

Tile will have been refreshed.

Related information

- [Changes not reflected when brush is modified on page 119](#)

Refreshing all Painted Tiles

Sometimes it is necessary to refresh previously painted tiles when their associated brush or linked prefab has been modified. Fortunately this can be resolved by refreshing the tiles.

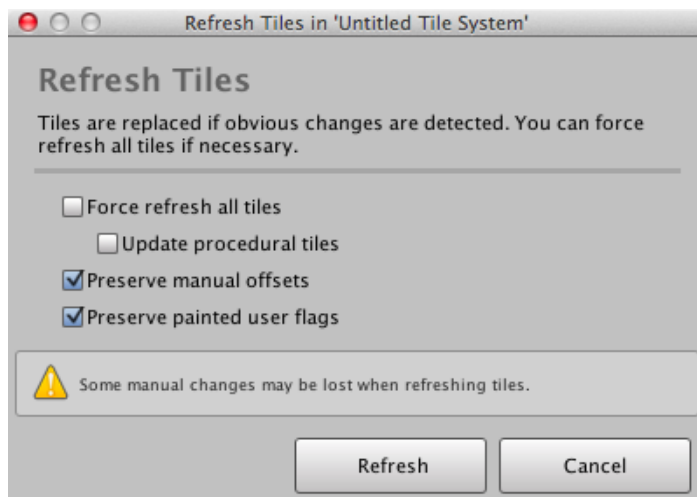
Before you begin

Ensure that **Inspector** window is shown by selecting **Window | Inspector**.

Procedure

1. Select your tile system.
2. Click **Refresh...** button in tile system inspector.

You should then see the following window:



3. Review and tick/untick the provided options:
 - **Force refresh all tiles** - Repaints all tiles using the brush that was previously used to paint them. Components and properties that have been manually tweaked will be reset to their original state.
 - **Update procedural tiles** - Regenerates procedural meshes.
 - **Preserve manual offsets** - Attempts to preserve manually tweaked position, rotation and scale for tiles that are repainted.
 - **Preserve painted user flags** - Attempts to preserve current flag states for tiles that are repainted. This option must be deselected to reflect changes that have been made to brushes.
4. Click **Refresh**.

Related information

- [Changes not reflected when brush is modified on page 119](#)
- [Updated tileset not reflected on page 126](#)

Chapter 5.

Brushes

A brush is an asset which can be created and designed using the Rotorz Tile System interface and then used to paint tiles onto tile systems. There are several types of brush each with the ability to paint different kinds of tiles.

Related information

- [Brush Designer on page 13](#)

Designing Brushes

Brushes are designed using the brush designer window which allows you to define several types of brushes using an intuitive user interface.

Brush Categories

Brushes can be organized into categories to make them easier to browse when painting.

Categories can be grouped into popout menus by specifying path style labels; for instance, "Props/Collectables". This can make it easier to browse and select categories when working with lots of categories.

Related information

- [Filter Menu on page 11](#)

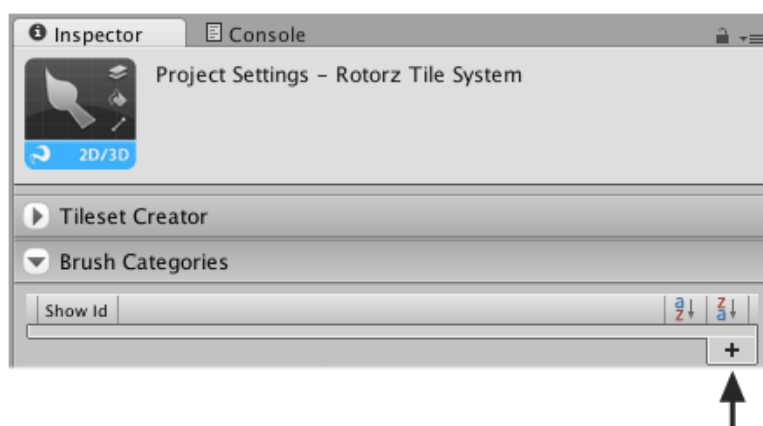
Adding a Brush Category

Brush categories can be added using the Rotorz Tile System project settings asset using the **Inspector**.

Procedure

1. Select menu **Edit | Project Settings | Rotorz Tile System**.

You should then see the "Project Settings" configuration of Rotorz Tile System in the **Inspector**:



2. Input a label for the category.



Removing a Brush Category

Brush categories can be removed using the Rotorz Tile System project settings asset using the **Inspector**.

About this task

Any brushes placed within a category that is removed will remain in the category, but will only be accessible when using the **Show All** filter.

Procedure

1. Select menu **Edit | Project Settings | Rotorz Tile System**.

You should then see the "Project Settings" configuration of Rotorz Tile System in the **Inspector**.

2. Click **Remove** button to the right of the category that is to be removed.



Common Brush Properties

Some properties are common between brush types which are located at top of brush designer interface.

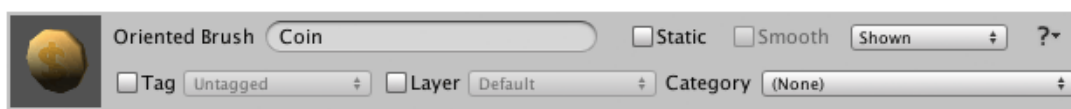


Figure 5-1. Common properties in brush designer

Brush Name

Used when naming the brush asset and is presented in brush lists.



Tip

Do not forget to click the **Rename** button once you have finished altering the brushes name.

Static

Indicates whether tiles painted using brush can be optimized when building tile systems.

See [Optimization on page 36](#) for more information.

Smooth

Indicates whether normals of tiles painted using brush should be smoothed when optimizing tile systems. This can help to avoid lighting artifacts where tiles join one another.



Note

Smoothing can only be enabled for static brushes.

Visibility

Allows brushes to be hidden or favored within brush list views, most specifically within the **Brushes** palette.

- **Shown** - Shown by default.
- **Hidden** - Hide from brush lists (except designer window).
- **Favorite** - Favorite brushes are shown plus can be filtered, and favored tileset brushes will appear under the **Brushes** list view.

Tag and Layer

Specify tag and layer that should be assigned to painted tiles.

These properties must be [explicitly overridden on page 77](#) for certain brush types. With oriented brushes this is useful because it allows the user to decide whether to override values for all tiles, or alternatively allows the user to define values on a per tile basis.



Figure 5-2. Example of tag override for alias and oriented brushes

Category

Categories can be used to filter brushes to help reduce the need to scroll through large tile sets.

You can define categories using the category manager which can be accessed via by selecting **Manage Categories** from the popup menu (see [Brush Categories on page 61](#)).

Extended Properties

Some extended properties are common between brush types (like group number and flags) whilst other specialized properties are specific to certain brush types.

By default the extended properties section is hidden. To display these properties click the **Extended Properties** tab along right of brush designer window as annotated in [Figure 5-3. Expand extended properties in brush designer on page 64](#).

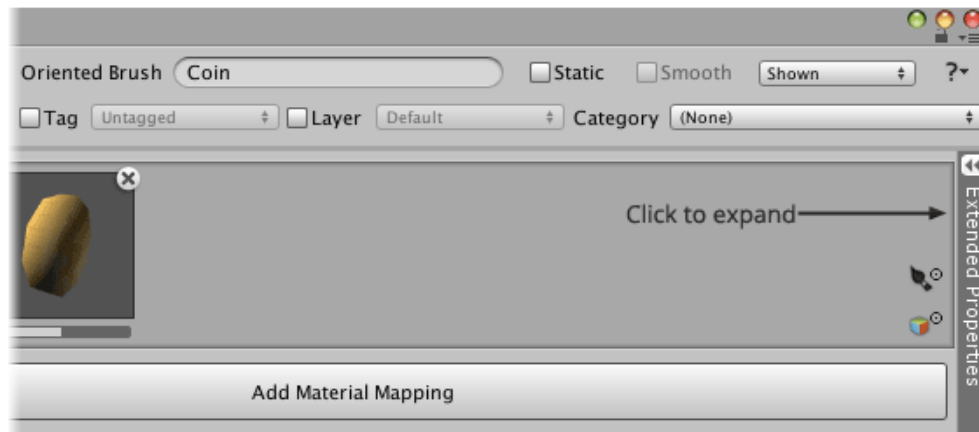


Figure 5-3. Expand extended properties in brush designer

Group No.

Brushes can be grouped which is useful when brushes are coalesced by group. Users may decide to utilize group numbers for other purposes instead.

For example, tiles painted using an oriented or autotile brush can coalesce with other tiles within a specific group.



Note

Group number is a 32-bit signed integer.

Force Legacy Sideways

Forces the legacy behavior of tile systems with sideways facing tiles for tiles painted using brush. With the legacy behavior tiles are rotated by 180 degrees around the up axis so that they face away from tile systems.

The behavior of tile systems with sideways facing tiles was changed in Rotorz Tile System version 2.2.2 to resolve workflow issues with Unity 4.3 sprites.

Whilst the new behavior is recommended for new projects, it is perfectly fine to continue to use the legacy behavior. Existing brush assets are automatically configured for the legacy behavior upon updating from an older version of Rotorz Tile System (pre v2.2.0) so that they behave consistently.

Scale Mode

Defines the way in which painted tiles should be scaled.

Scale Mode	Description
Dont Touch <i>(default for new brushes)</i>	Leave tile alone! Maintain scale of prefabs.
Use Tile Size	Apply scale from tile size to prefabs.
Custom	Apply custom scale to prefabs.

Always Add Container

Empty container game objects can be forced for certain brush types even when container objects are not actually required. This can be useful for custom scripts though should be avoided unless necessary.

Container objects are named "tile" and are generated by tileset brushes when game objects are attached (for example, attached prefabs).

Flags

Flags can be used in conjunction with user defined scripts and are intended for general usage. There are 16 unnamed flags plus one called "solid".

Initial state of tile flags can be specified using the brush designer. Flag states are maintained on a per tile basis and are initialized from brush upon being painted. Tile Flags can be accessed and modified at runtime by custom scripts if needed (see [TileData](#)).

In addition to the 16 general purpose flags an additional flag labelled "Solid" is also provided which is exposed as a named property. The solid flag is also a general purpose flag which indicates whether a tile should be treated as solid or passable. Custom scripts can be implemented to utilize the state of this flag to perform tile based collision detection or pathfinding.



Figure 5-4. Flag properties in brush designer



Note

Flag states are stored within a single 32-bit integer variable and accessed using bit masks. Additional flags are used internally and are not directly accessible using the API.

You can provide your own labels for the 16 general purpose flags which can be specified for all brushes project-wide or overridden on a per brush basis.

Offset, Rotate and Scale Prefabs

Sometimes you may find that your tiles are do not align in the way that you had expected. The way in which prefab instances are positioned can be influenced using the transform properties of tile prefabs themselves.

Whilst it is generally better to fix such errors using your modelling software, it is often more convenient to solve this using the prefab offset.

For example, the tile offset can be specified by altering the prefab position property using the **Inspector** window. Rotation and scaling can be altered in a similar way:

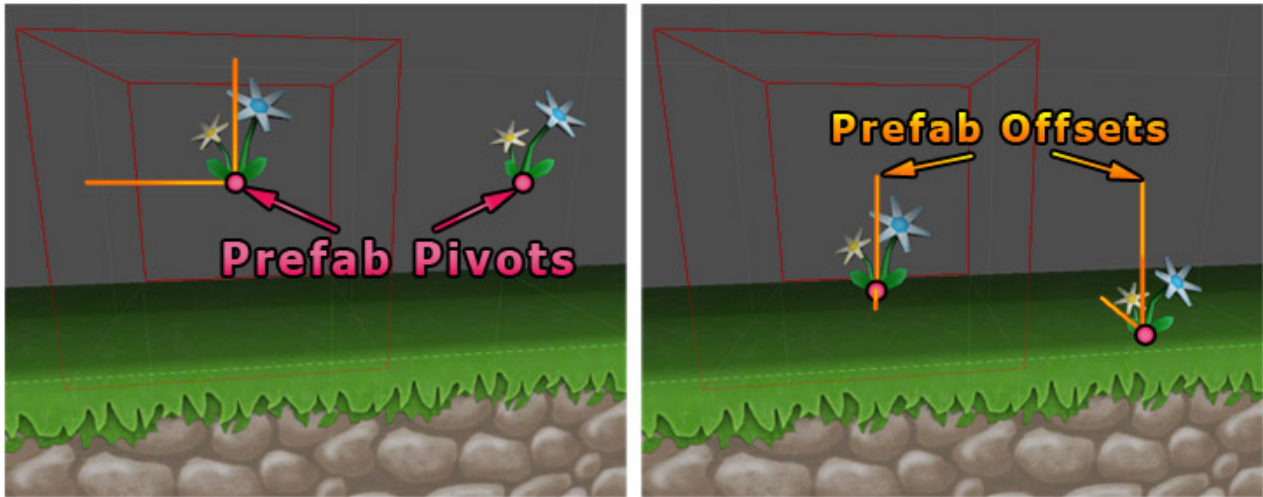


Figure 5-5. Illustration of tile offsets and how they can be applied




Important

Ensure that **Apply Prefab Transform** is set for brush using brush designer.

Use as Prefab Offset

This utility function aims to make it easier to define prefab offsets for your brushes:

1. Paint tile into the desired grid cell (even though it looks wrong).
2. Select tile prefab instance.
3. Modify position, rotation and scale as needed.
4. Select menu  | **Use as Prefab Offset**.

Remapping Materials

The materials of painted tiles can be remapped making it possible to reuse existing brushes but with different materials. This can be a massive time saver when working with oriented brushes.

Whilst the concept of remapping materials can be applied to other brush types, it is most applicable for alias brushes. In fact remapping materials is one of the primary purposes of alias brushes. Multiple mappings can be specified for brushes that refer to objects that use multiple materials.

Figure 5-6. [Material mapper user interface on page 67](#) illustrates how the default material of the "Smooth Platform" brush can be remapped to "Grass Stone" in the "Grass Block" alias brush. The same "Smooth Platform" master brush can also be remapped to other materials (like "Cave" for example).

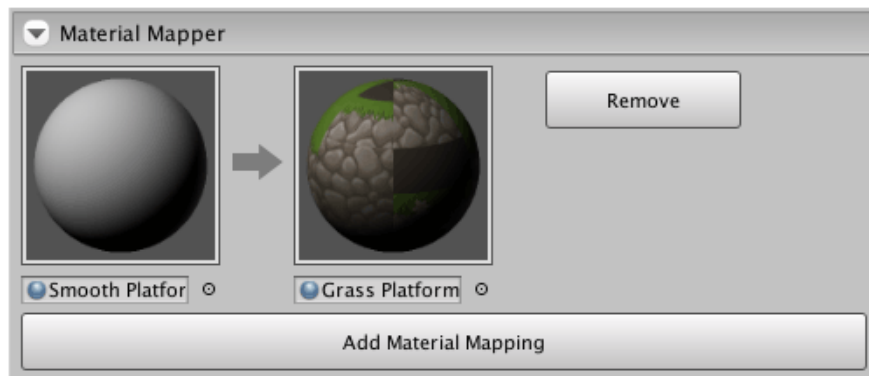


Figure 5-6. Material mapper user interface



Tip

The **Add Material Mapping** button will attempt to find the next material to remap. Materials are automatically selected from the default orientation of oriented brushes.

See [Master Brushes on page 81](#) for further information regarding master brushes.

Oriented Brushes

General purpose brush type that can be used to paint singular objects or automatically orienting platforms, walls, etc. Oriented brushes automatically choose and paint tiles based upon the surrounding tiles.

What is an Orientation?

The orientation of a tile is defined by the tiles that surround it which allows oriented brushes to make automatic selections.

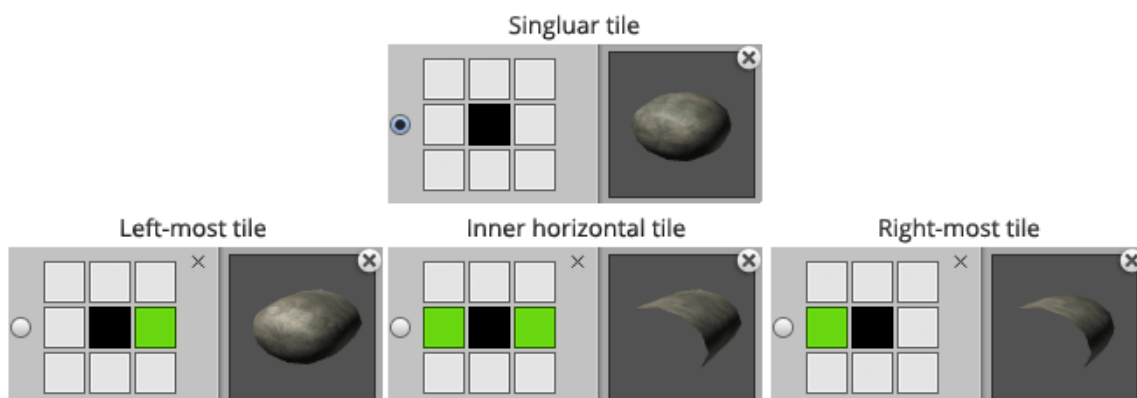


Figure 5-7. Orientations of a horizontal platform brush

One or more tile variations can be specified for each defined orientation. [Figure 5-8. Multiple egg variations on page 68](#) illustrates the oriented brush that was used to randomly paint eggs when designing levels for *Munchy Bunny!*



Figure 5-8. Multiple egg variations



Note

"Egg" is an oriented brush where only the default orientation is used.

Variation Weights

Chance of variations being picked when painting with randomization can be adjusted using weight slider shown immediately below variation preview in oriented brush designer.

Variations which share the same weight have an equal chance of being picked when using randomization. For instance, both of the following examples provide the same behavior:



Figure 5-9. Two equivalent examples of variation weights

In the following two equivalent examples the coin variation should be picked twice for each diamond which is picked (on average) when painting:

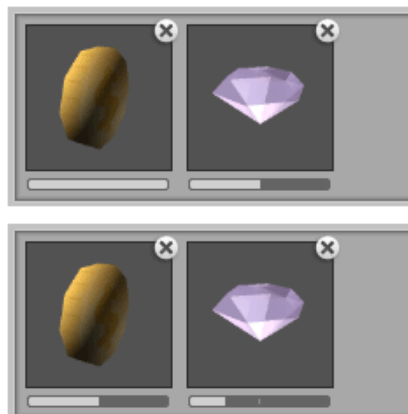


Figure 5-10. Coins are more common with variation weights

Adjust Variation Weight for Entire Column

Sometimes it is useful to adjust weight for an entire column of variations within an oriented brush; for instance, the first variation within each orientation.

To achieve this hold **Ctrl** whilst dragging variation slider in any orientation.

Exclude Variation From Randomization

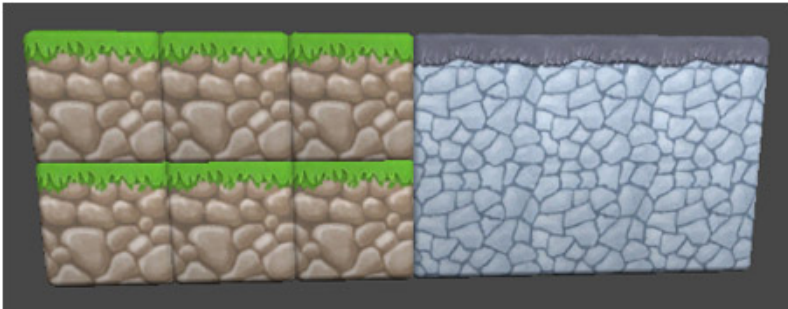
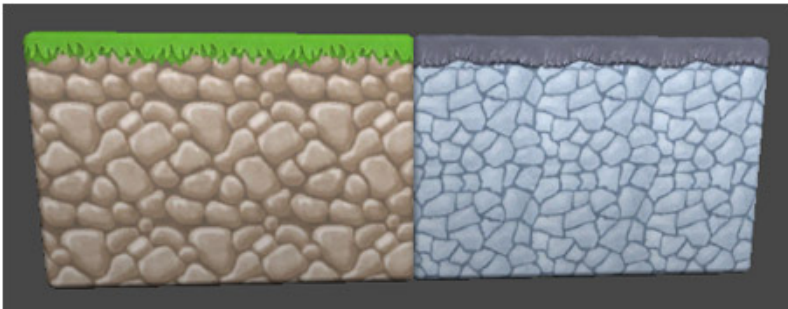
Specify a weight of zero to remove variation from being randomly picked. First variation is assumed when all available variations have a weight of zero. For instance, the following will always paint diamond variations when randomization is being used:



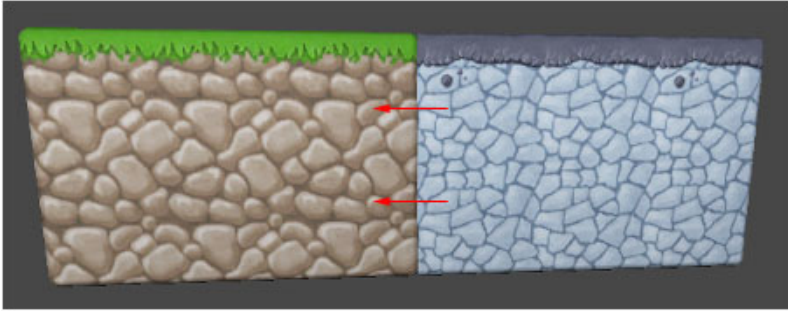
Figure 5-11. Avoid picking variation at random with zero weight

Coalesce Mode

The way in which tiles join with one another can be controlled by altering the way in which they coalesce.

Coalesce Mode	Description
None	<div>Do not attempt to join adjacent tiles.</div> <div></div> <div><ul style="list-style-type: none">• Grass Block with Coalesce Mode None• Cave Block with Coalesce Mode Own</div>
Own <div>(Default for new brushes)</div>	<div>Only join adjacent tiles of same brush.</div> <div></div> <div><ul style="list-style-type: none">• Grass Block with Coalesce Mode Own• Cave Block with Coalesce Mode Own</div>

Coalesce Mode	Description
Other	<p>Do not join with adjacent tiles of own brush, but join with tiles that were painted using any other brush.</p>  <ul style="list-style-type: none">• Grass Block with Coalesce Mode Other• Cave Block with Coalesce Mode Any <div>Note<p>Red lines illustrate that brushes join with one another.</p></div>
Any	<p>Join with adjacent tiles of own brush and any other brush.</p>  <ul style="list-style-type: none">• Grass Block with Coalesce Mode Any• Cave Block with Coalesce Mode Any
Groups	<p>Only join with adjacent tiles that were painted using brush of zero or more specific groups.</p>  <ul style="list-style-type: none">• Grass Block with Coalesce Mode Groups (in group #1 with group #1)• Cave Block with Coalesce Mode Groups (in group #1 with group #1)

Coalesce Mode	Description
Own And Groups	<p>Join with adjacent tiles that were painted using same brush or brush of zero or more specific groups.</p>  <ul style="list-style-type: none"> • Grass Block with Coalesce Mode Own (in group #1) • Cave Block with Coalesce Mode Own And Groups (with group #1)

Fallback Orientation

Default orientation is assumed when actual orientation of tile is not defined within brush (or next best was not matched). The default orientation is also used to generate preview for brush.

Fallback Orientation	Description
Next Best <i>(default for new brushes)</i>	Attempt to find next best orientation that is available before assuming the default orientation.
Use Default	Use default orientation when no strong matches are found.
Use Default Strict	Use default orientation when no exact matches are found.

Related information

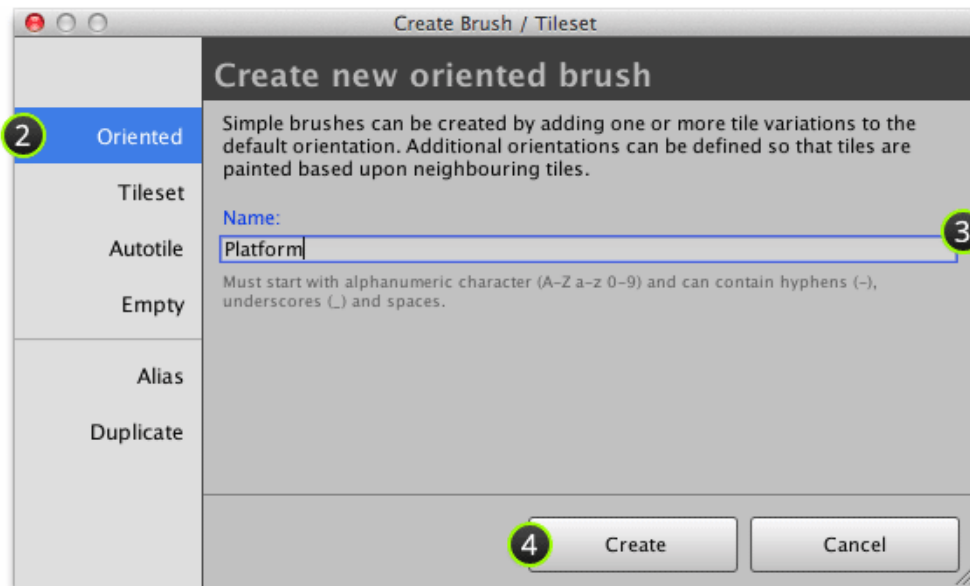
- [Set Default Orientation on page 73](#)

Creating an Oriented Brush

Oriented brushes can be created using the create brush window.

Procedure

1. Select menu command  | **Create Brush or Tileset...**
2. Ensure that **Oriented** tab is selected:

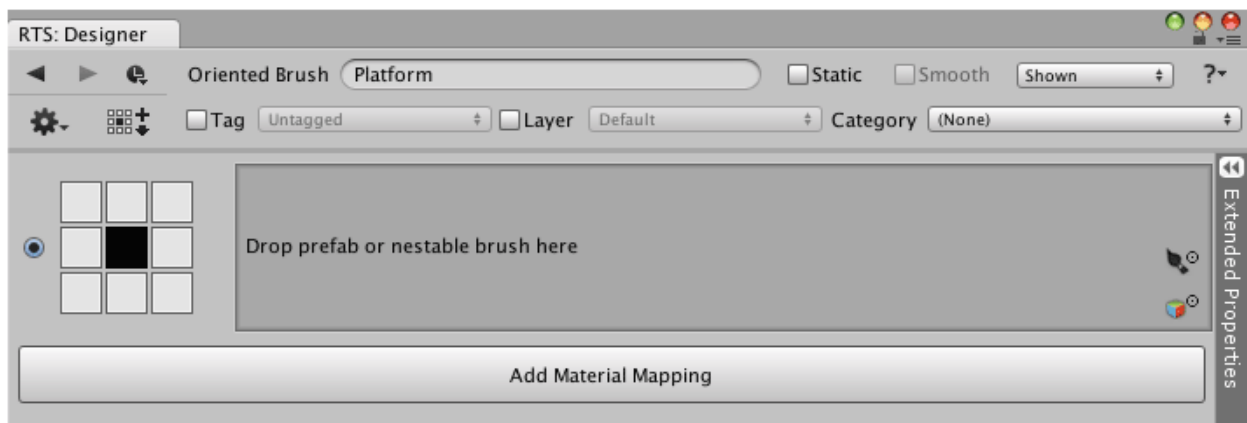


3. Input *unique* name for brush.

4. Click **Create**.

Results

You should then see something like the following:

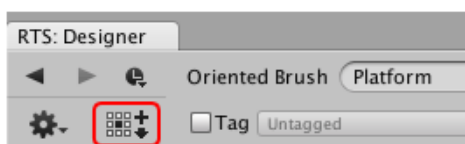


Define or Find Orientation

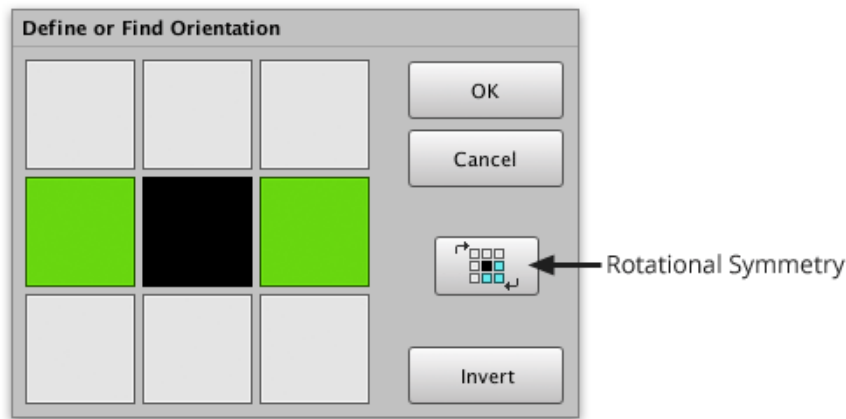
New orientations can be defined by specifying the expected neighboring tiles by toggling them green or white. The same interface can also be used to quickly find existing orientations which is useful when working with brushes which have lots of them.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Click **Define or Find Orientation...** (F3) button.



3. Select orientation by clicking the squares which surround the central black square to toggle between white and green. Quickly toggle multiple squares by dragging mouse pointer across them.
 - **Black Square** - The tile that you are matching.
 - **Green Square** - Neighboring tile which coalesces with tile.
 - **Cyan Square** - Same as green but indicates rotational symmetry is selected.
 - **White Square** - Empty or neighboring tile which does not coalesce with tile.

**Tip**

Use **Invert** button to quickly toggle all squares with one click. This can be a time saver when orientation is mostly green!

4. Click **OK** to define new orientation.

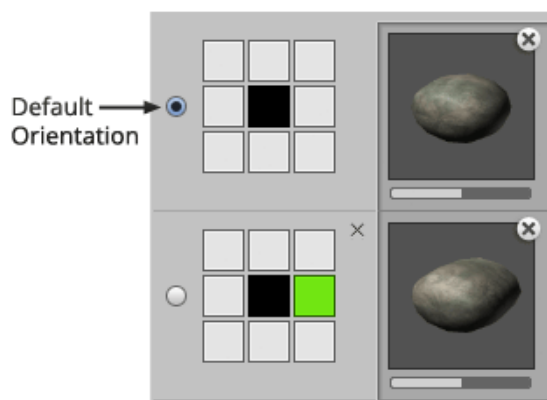
Brush designer will automatically scroll to orientation if it already exists. It is not possible to define the same orientation multiple times.

Set Default Orientation

Each oriented brush has a default orientation which is marked using the radio button towards the left.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Click radio button at left of orientation to mark as default.



Related information

- [Fallback Orientation on page 71](#)

Remove Orientation

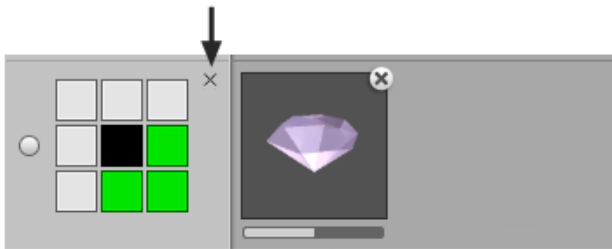
Unwanted orientations can be removed by clicking the small cross button shown to right of 3x3 icon.

About this task

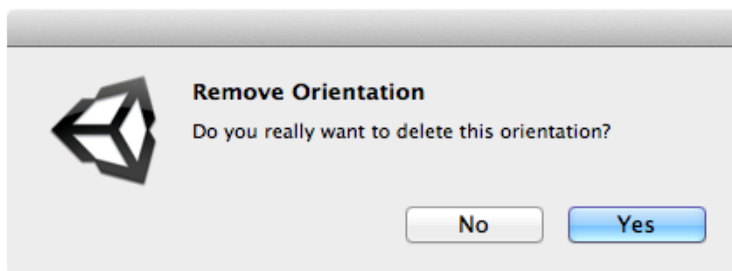
The default orientation cannot be removed. If the default orientation is not wanted then you must first mark another orientation as the default.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Click **Remove Orientation** cross button on the orientation that you want to remove.



A confirmation message should appear:



3. Select **Yes** to remove orientation.

Add Variation

Multiple variations can be added to each orientation when desired. There are two kinds of variation; tile prefabs and nestable brushes. The most common form of nestable brush is the tileset brush.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Click circle at lower right of orientation to select one or more variations to add.

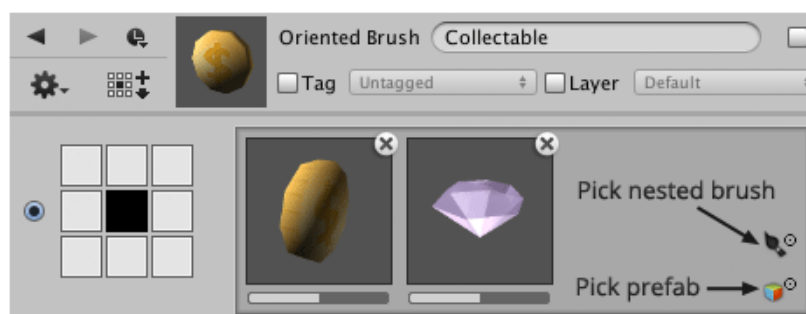


Figure 5-12. Example of brush that randomly paints coins and diamonds

When object browser is visible multiple variations can be added by single clicking the ones that you would like to add. Double-clicking will add a single variation and close object selector window.

**Tip**

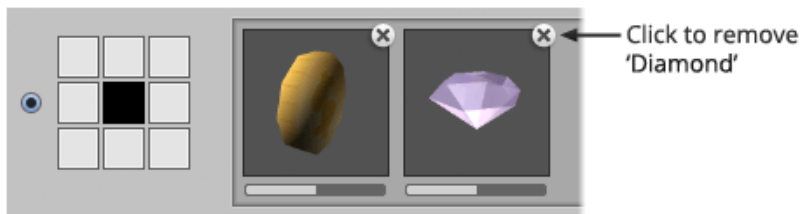
Prefabs can also be dragged from the **Project** window and brushes can be dragged from the **Brush** palette and then dropped onto the orientation.

Remove Variation

Tile variations can be removed from an orientation by clicking the small cross button using the brush designer.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Click small cross button at top right of variation.

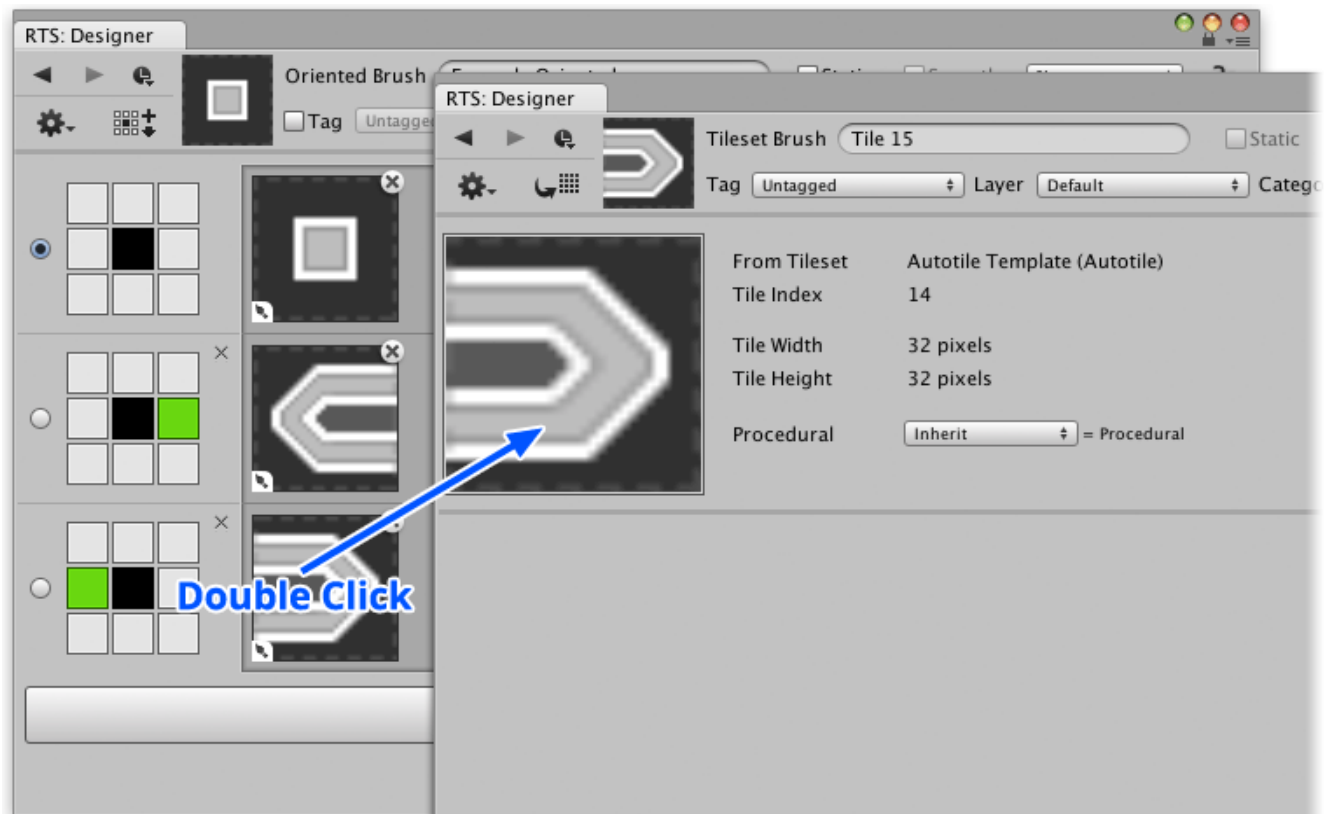


Looking up a Variation

This section explains how to quickly lookup a tile prefab or edit a nested brush when using the brush designer, which can be a huge time saver when continuously switching to and fro between an oriented brush and its nested brushes.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Double click the thumbnail of the variation that you would like to lookup.



Results

The outcome will vary depending upon variation is prefab or nested brush:

- **Tile Prefab** - Highlight in project window.
- **Nested Brush** - Show nested brush in designer.

Reorder Variation

Variations can be reordered by dragging them around one another or even across to another orientation.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Position mouse pointer over variation.
3. Press and hold left mouse button.
4. Drag variation into its new position.
5. Release mouse button to accept new position; or press the **Esc** key to cancel.

Copy Variation

A variation can be effortlessly copied to another orientation by holding the **Ctrl** key whilst dragging it.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Position mouse pointer over variation.
3. Press and hold left mouse button whilst holding the **Ctrl** key.
4. Drag variation into another orientation.
5. Release mouse button to copy variation and insert into position; or press the **Esc** key to cancel.

Alias Brushes

Alias brushes can be created to define new brushes based upon existing brushes. Materials can be remapped allowing you to reskin the target brush. It is also possible to override the values of certain properties.

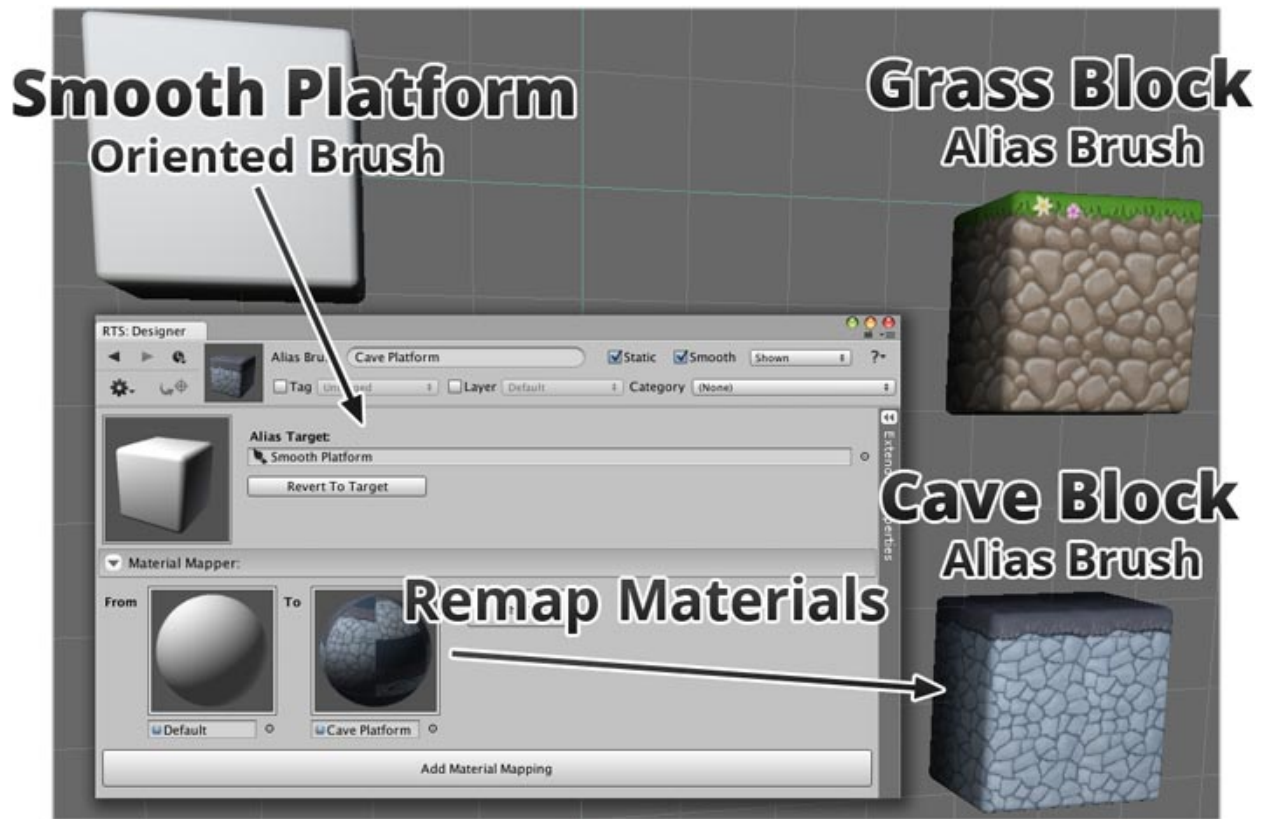


Figure 5-13. Demonstration with aliases of "Smooth Platform" brush

Related information

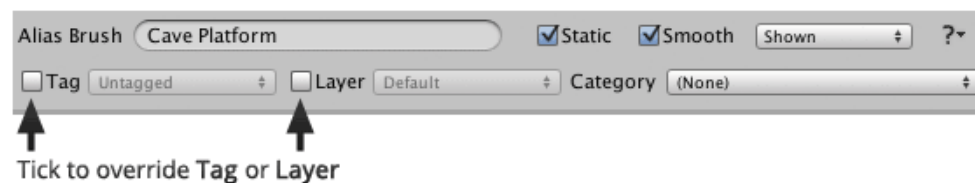
- [Remapping Materials on page 66](#)

Overriding Properties

Alias brush designer interface varies depending upon the type of brush which is being targeted making it possible to override specialized aspects of target brush.

Common Properties

Common properties can be specified as with most brush types, though as shown below, some properties must be explicitly overridden by ticking an extra box. This allows you to selectively override the values of certain properties whilst inheriting the values of others.

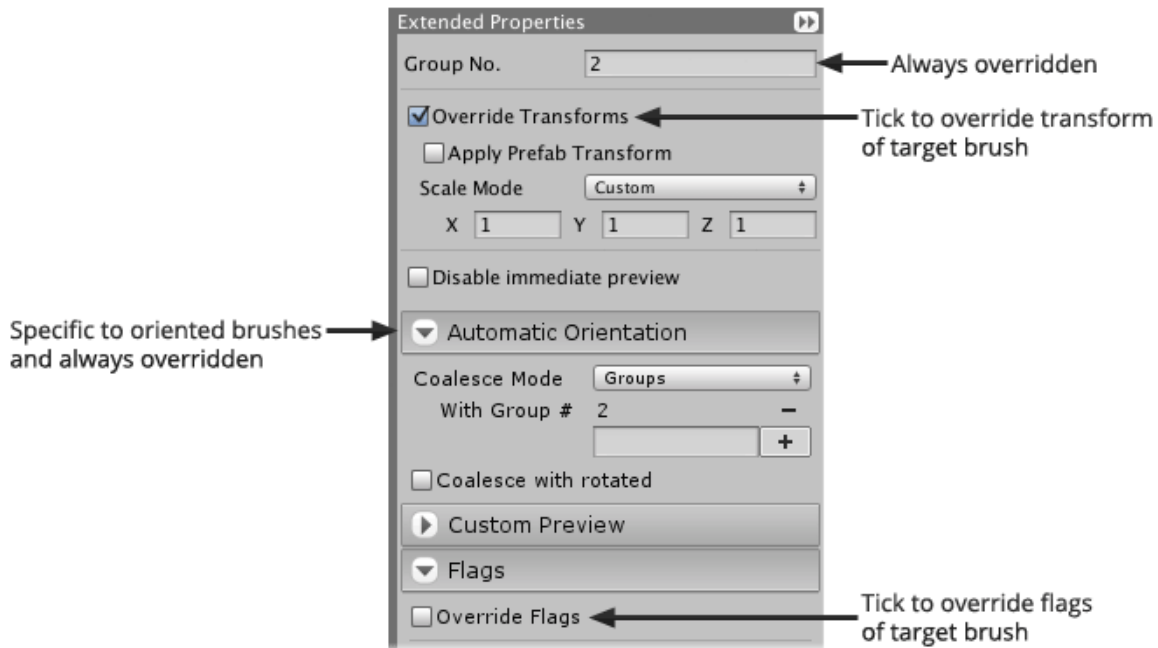


**Note**

The value of inherited properties reflects changes made to target brush which is shown in greyed out fields in designer interface.

Extended Properties

The range of extended properties presented will vary depending upon the type of target brush. For example, some properties apply to all brush types whilst others only apply to certain types of brush.

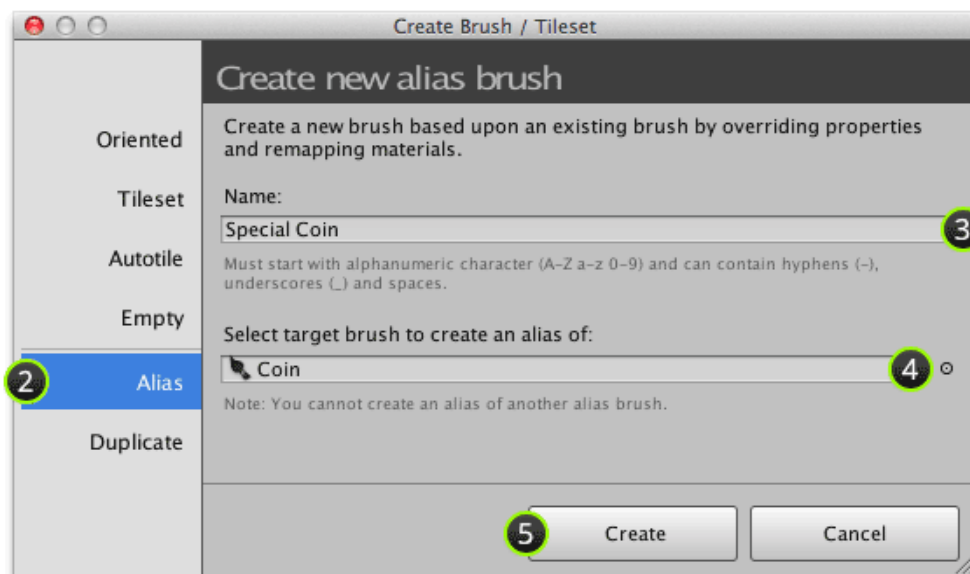


Creating an Alias Brush

Alias brushes can be created using the create brush window which can be quickly accessed via context menu of target brush.

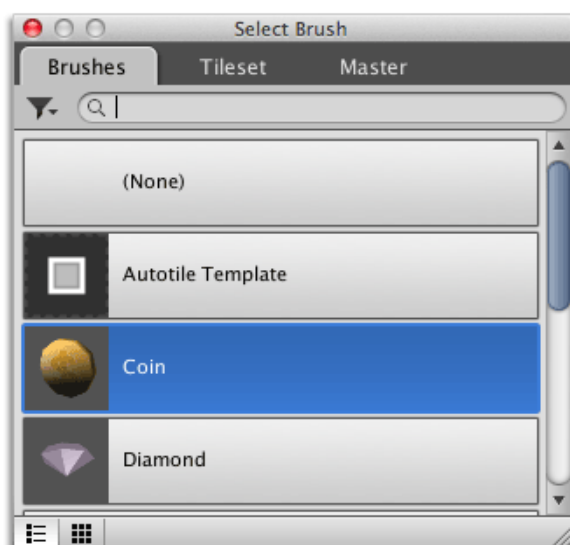
Procedure

1. Select menu command | **Create Brush or Tileset...**
2. Select **Alias** section:



3. Input *unique* name for brush.
4. Select target brush to create alias of.

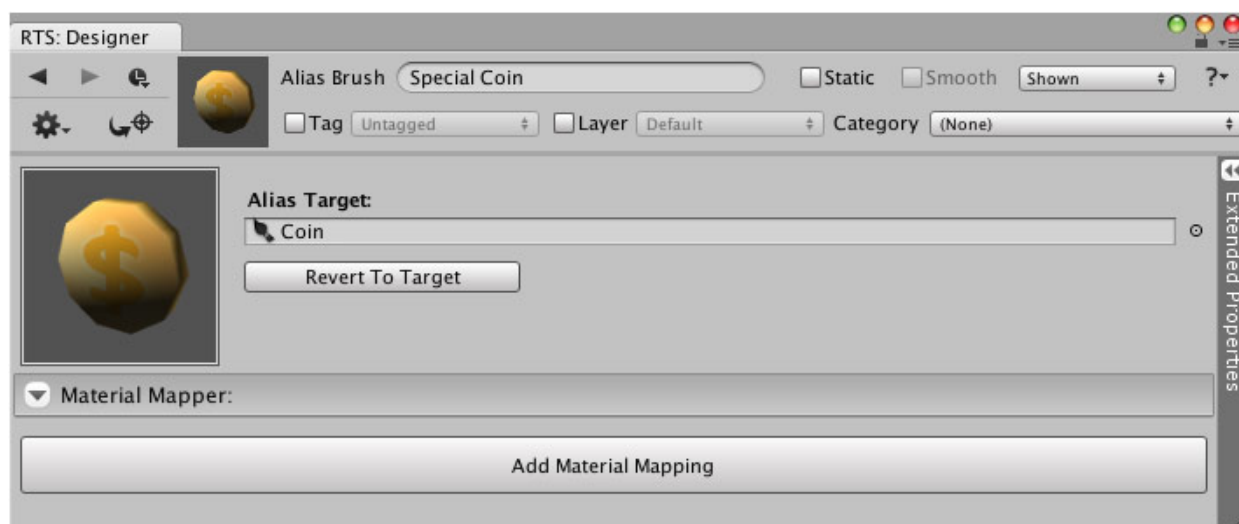
To do this either click brush field and select brush as demonstrated below or drag and drop brush from brush list.



5. Click **Create**.

Results

You should then see something like the following:



Empty Brushes

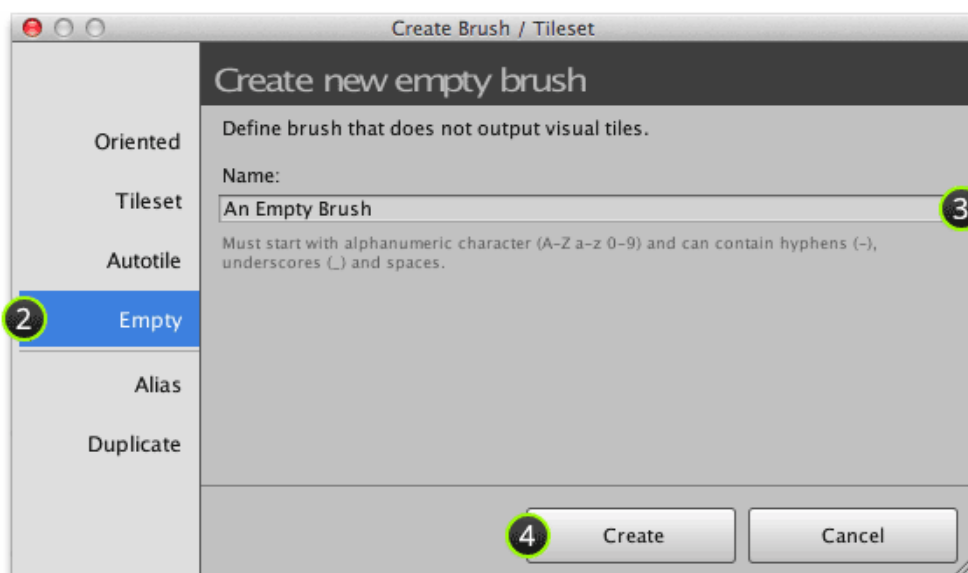
Tiles painted using empty brushes have no visual representation which can be useful to fill spaces within oriented brushes, or to assist with custom game logic. For example, it is possible to specify flags.

Creating an Empty Brush

Empty brushes can be created using the create brush window.

Procedure

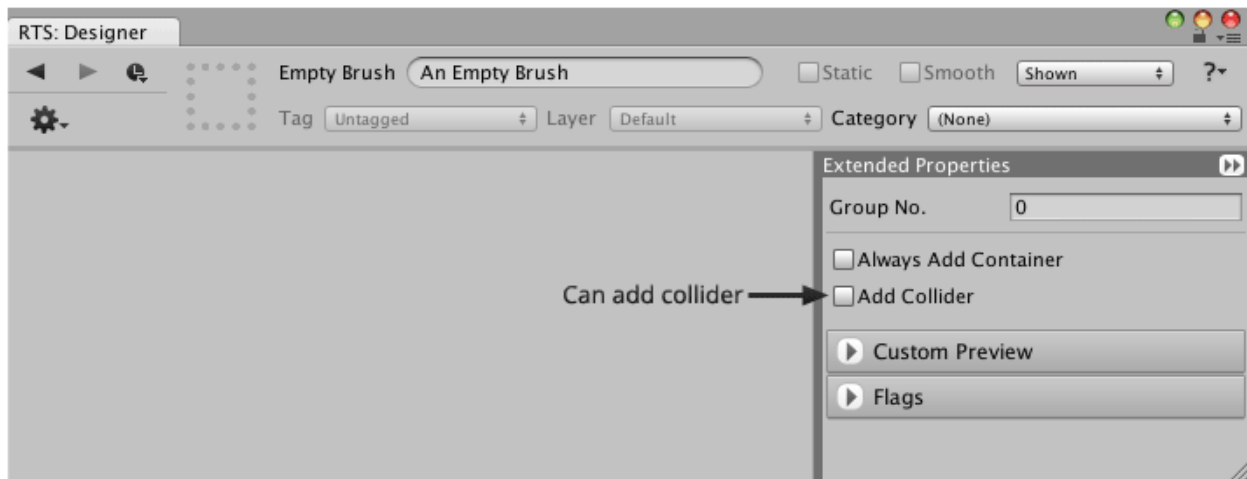
1. Select menu command  | **Create Brush or Tileset...**
2. Select **Empty** section:



3. Input *unique* name for brush.
4. Click **Create**.

Results

You should then see something like the following:



Master Brushes

Master brushes are normal brushes except they cannot be edited using designer without first moving them into the "TileBrushes" folder in your project. The purpose of master brushes is to define reusable templates which cannot be inadvertently modified.

For example, the provided "Smooth Platform" master brushes can be reused by creating new alias brushes like the "Grass" and "Cave" ones provided. Master brushes can be duplicated when greater control is needed (i.e. to add/remove orientations and variations).

Related information

- [Creating an Alias Brush on page 78](#)
- [Duplicating a Brush on page 82](#)

Creating a Master Brush

A regular brush can be turned into a master brush by moving it's associated asset into a "TileBrushes/Master" folder somewhere within your project.

Before you begin

Whilst it can be useful to turn oriented brushes into highly reusable master brushes, it is rarely useful to apply this notion to alias or tileset brushes.

Procedure

1. Locate brush asset within your project.

Your brush will be located somewhere in the `TileBrushes` folder.



Tip

Quickly locate brush asset by right-clicking brush in **Brush** palette and then selecting **Reveal Asset**.

2. Create a folder called "Master" inside "TileBrushes" folder using the **Project** window.

For example:


- `Assets/TileBrushes/Master`

- **Assets/YourFolder/TileBrushes/Master**

3. Drag brush asset into "Master" folder.


CAUTION

You may encounter problems if you attempt to move your brush asset outside of the Unity interface (i.e. using Explorer or Finder).

4. Select menu command  | **Rescan Brushes**.

Results

Your brush should then be listed as a master brush.

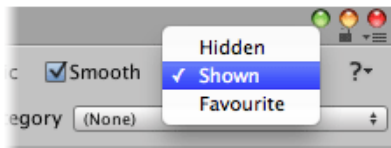
Hiding a Brush

Brushes can be hidden from the brush palette by altering its visibility using the brush designer. This is useful for intermediary brushes which are not intended for direct usage.

Procedure

1. Right-click brush using **Brush** palette and select **Show in Designer...** from menu.
2. Select **Hidden** from **Visibility** field.

Field is located in upper right corner of designer interface:

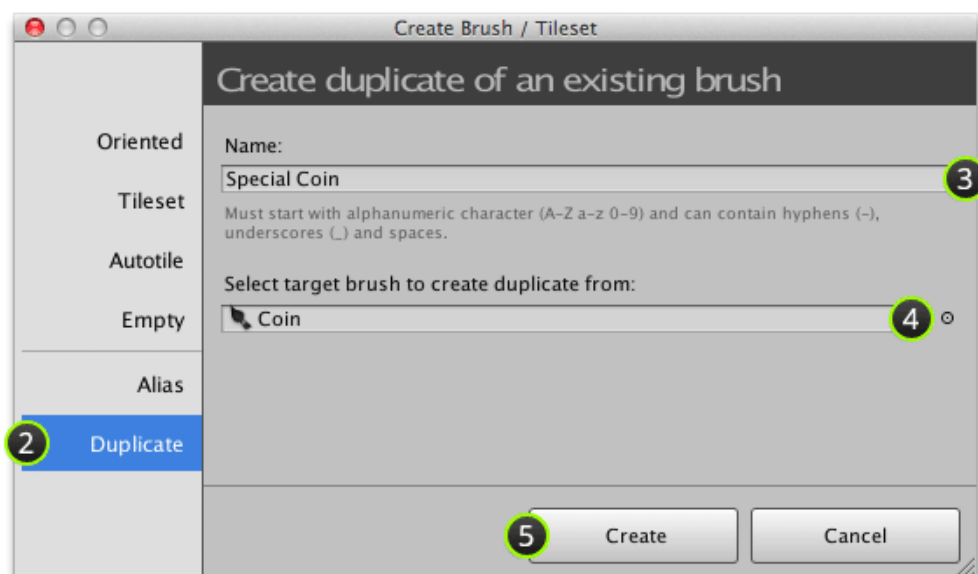


Duplicating a Brush

An existing brush can be duplicated using the create brush window which can be quickly access via the brush's context menu.

Procedure

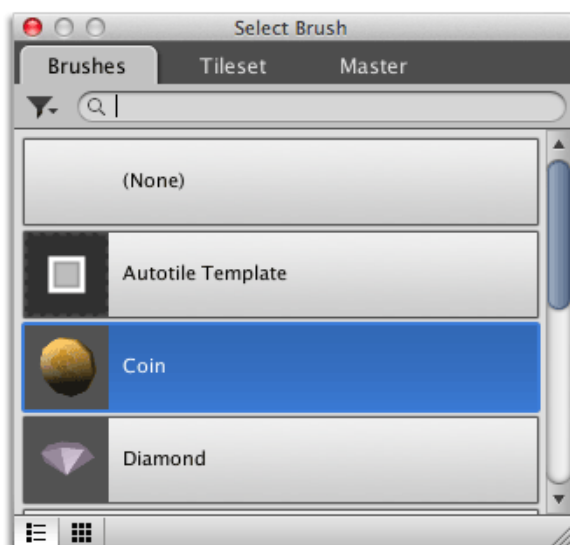
1. Select menu command  | **Create Brush or Tileset...**
2. Select **Duplicate** section:



3. Input *unique* name for brush.

4. Select the target brush to create duplicate from.

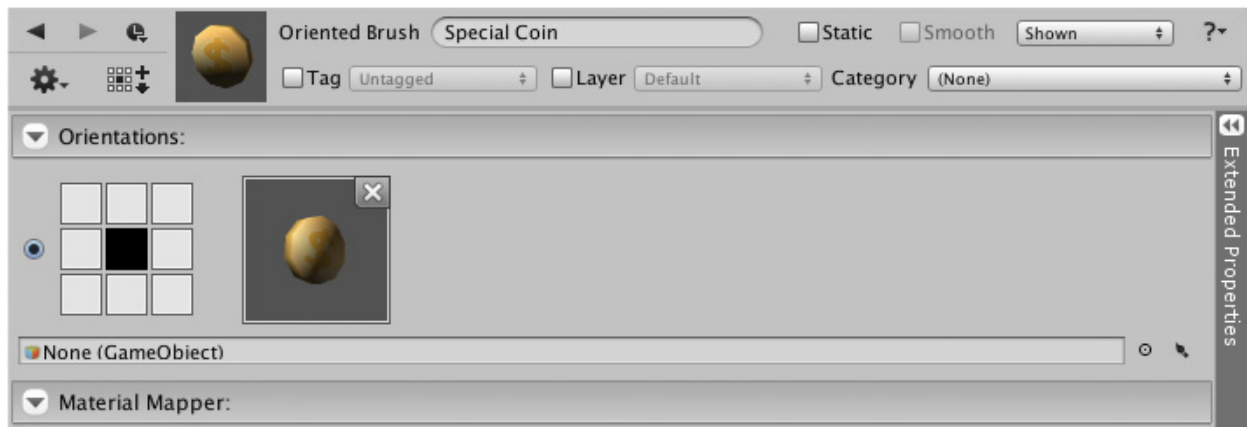
To do this either click brush field and select brush as demonstrated below or drag and drop brush from brush palette.



5. Click **Create** to duplicate target brush.

Results

You should then see something like the following:



Important

Ensure that duplicate brush is shown in designer before making any changes.

Deleting a Brush

An unwanted brush can be deleted via its context menu in the brush palette or via the brush menu when brush is shown in designer.

Before you begin

Ensure that **Brush** palette is shown by selecting  | **Editor Windows** | **Brushes**.

Avoid deleting brush if still referenced by one or more tile systems to avoid causing unintentional damage. It will no longer be possible to refresh existing tiles that were painted using brush once deleted. Inability to refresh tiles prevents automatic orientation and can cause issues when optimizing tile systems.



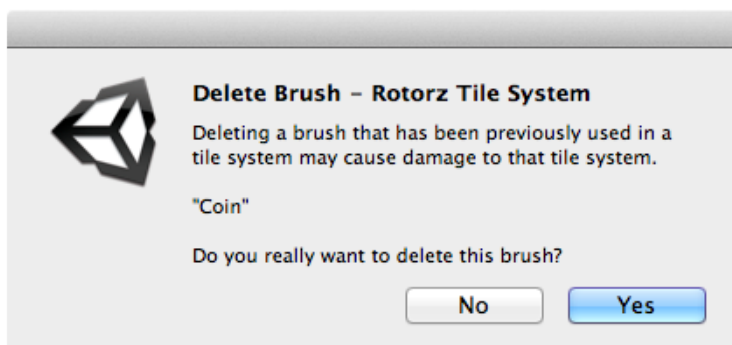
Note

Master brushes can be deleted using the **Project** window.

Procedure

1. Right-click on the brush that you would like to delete.
2. Select **Delete Brush...** from context menu.

A message box will appear to confirm deletion of brush:



3. Select **Yes** to delete brush.

Chapter 6.

Tilesets

Tilesets make it easier for you to define brushes that allow you to paint 2D tiles onto your tile systems. You can define a tileset from an atlas texture which includes the artwork for a number of uniformly sized tiles.

Each tileset is represented with an asset which:

- Associates the tileset with its atlas texture and material.
- Defines the way in which tiles are packed.
- Contains tileset brushes that are used when painting tiles.
- Manages pre-generated mesh assets for non-procedural tiles.

Once you have defined your tileset you can create a [tileset brush on page 96](#) for each of the tiles that you would like to be able to paint with. Tileset brushes can then be used with the provided editor tools or dynamically using the [Runtime API on page 106](#).



Remember

If you previously used version 1.x of this extension then you may have used atlas brushes for painting 2D tiles. These have been replaced with tilesets brushes which can be used procedurally or non-procedurally. Previously atlas brushes were only able to paint tiles non-procedurally.

Related information

- [Tilesset Designer on page 14](#)

Edge Correction

Bleeding will usually occur at the edges of tiles due to texture filtering and/or mip-mapping. Two popular ways to reduce the visibility of such artifacts is to either inset UV coordinates by half a texel, or to add borders around the edges of each tile in your atlas.

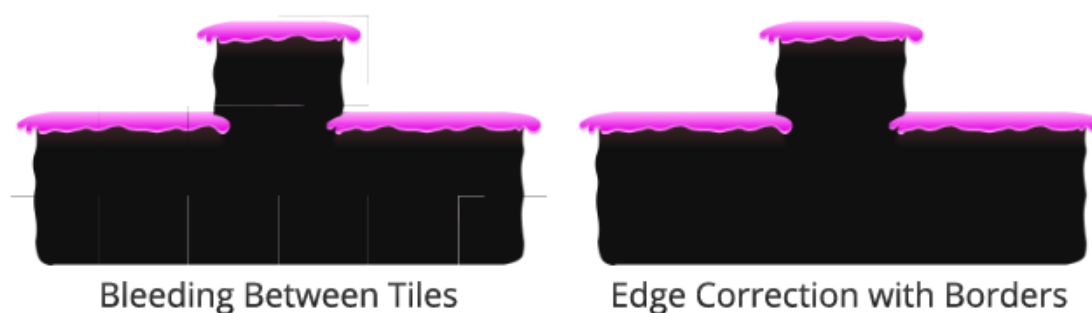


Figure 6-1. Example of bleeding between tiles due to texture filtering

As depicted above, lines appear between tiles which often flicker when movement occurs (camera movement, zooming, tile movement, etc). The effects of edge bleeding can sometimes be counteracted by carefully packing tiles adjacent to other similar tiles.

Why does bleeding occur?

Edge bleeding occurs when multiple tiles are packed into an atlas texture because pixels are incorrectly blended between adjacent tiles.

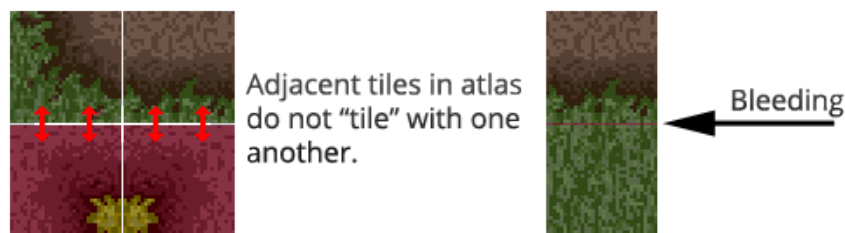


Figure 6-2. Example atlas (left) where edge bleeding occurs (right)

Each of the following can lead to this undesirable effect:

- Texture filtering
- Mip-mapping
- Lossy texture compression
- Use of Non-Power-Of-Two (NPOT) atlas texture
- Sub-pixel placement of tiles



Tip

Where possible avoid using lossy texture compression when working with tileset atlas textures. If you switch the texture inspector into advanced mode you can pick from a range of texture formats on a per platform basis.

Related information

- [Poor texture alignment in tileset brushes on page 125](#)

None

Specifies that edge correction should be disabled. This can be useful when blocky "pixel-like" graphics are desired when using point filtering, though edge correction is generally more effective.



Figure 6-3. Example of tiles without edge correction using point filtering

The following points may be helpful when using this option:

- Disable mip-mapping for texture.
- Use true color format (i.e. disable compression).

- Ensure that both tile system and camera positions are round numbers.
- Some people suggest adding 0.5 to rounded positions.

**Tip**

It is usually better to utilize one of the edge correction options to reduce the effects of bleeding at edges.

Borders

Adding borders around the edges of each tile is one of the most effective ways to reduce artifacts that are caused by bleeding. This works because the border pixels are blended with the tile pixels when filtering or mip-mapping is used.

The size of the border that you require will depend upon whether you are using mip-mapping for your textures. Larger borders allow you to view tiles from a greater range. A border size of between 1 and 8 pixels is usually adequate.

If your tile system is only ever visible to an orthographic camera, and its visual size never changes, then you can often just disable mip-mapping. Disabling mip-mapping will also reduce the amount of texture memory required.

**Tip**

Unlike with standard tilesets, borders can be generated automatically when creating an autotile tileset.

Adding borders for repeating edges:

Borders can be created by copying the pixels of the expected neighboring tiles. This technique has been applied to all 4 edges and corners for each of the tiles in the following illustration:

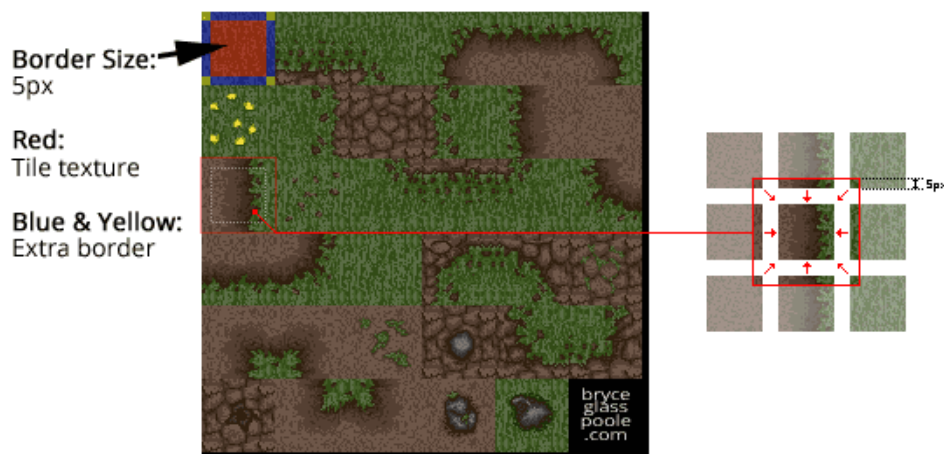


Figure 6-4. Adding border around the edges of each tile

Clamp non-repeating edges:

For the following tile you might feel tempted to place transparent pixels on its upper and left edges. Unfortunately doing so would still cause edge bleeding to occur resulting with a fine transparent line between that tile and the tile above (or to left) of it.

Instead the better solution is to clamp the edge pixels by stretching the edge pixels outward. In the case of this tile a combination of clamping and repeating is appropriate.

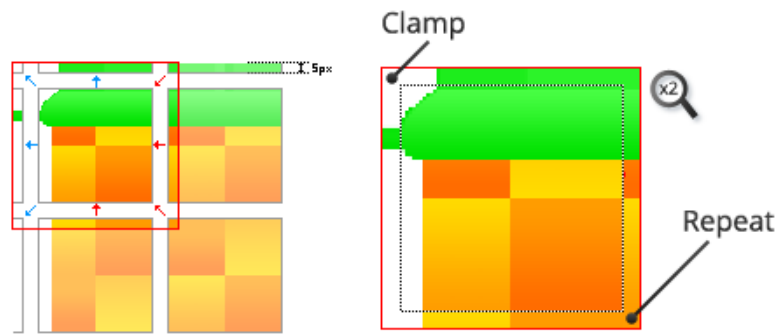


Figure 6-5. Clamping and repeating edges (Left: Composition, Right: Composed Tile)

Surround sprite-like tiles with transparent pixels:

Standalone sprite-like tiles can be surrounded by a border of transparent pixels:



Figure 6-6. Adding transparent borders to independent tiles

Delta / UV Inset

One of the most common ways to reduce bleeding at the edges of tiles is to inset tile UVs by a small amount; half a texel tends to be quite effective.

This technique works quite well for some scenarios. When tiles are viewed too closely you will notice that the outer pixels of each tile appear smaller than the other pixels. This is because the UVs have been inset by a small amount. It is likely that you will see tile borders when tiles are viewed closely with bilinear or trilinear filtering.



Tip

Where possible you should use the border correction method. Whilst it can take longer to produce the artwork, the results are generally better.

Procedural and Non-Procedural Tiles

Tiles that are painted using a tileset can be presented in one of two ways; procedurally or non-procedurally. The performance of painting and erasing procedural tiles tends to be better than that of non-procedural tiles whereas non-procedural tiles can offer greater flexibility.



Tip

You can paint a mixture of procedural and non-procedural tiles onto a single tile system if needed.

Procedural Tiles

Procedural tiles are presented using meshes that are procedurally generated on a per chunk basis. These can be painted and erased incredibly quickly and generally incur fewer draw calls than non-procedurally painted tiles.

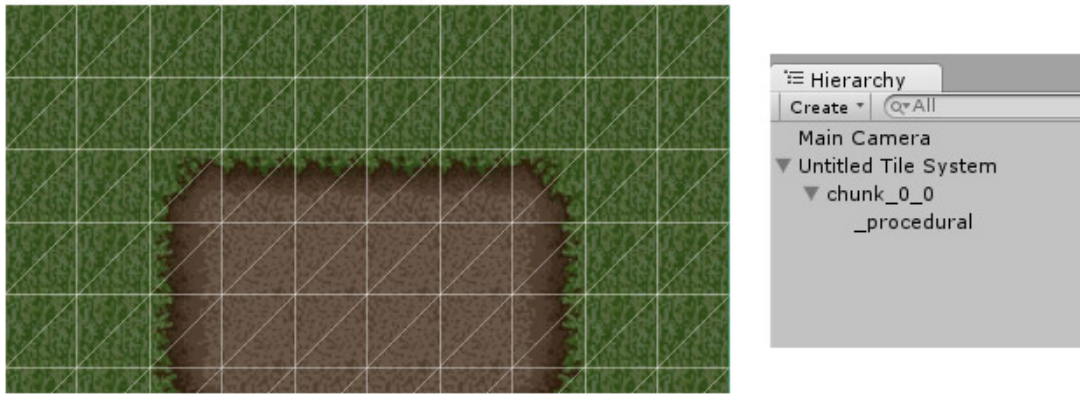


Figure 6-7. Tiles painted using procedural brush rendered using single draw call

Whilst game objects are not created for each tile, it is possible to attach game objects by specifying an attachment prefab, see [Attach Prefab on page 97](#). Colliders can also be added very easily, see [Add Collider on page 97](#).

Non-Procedural Tiles

Non-procedural tiles are individual game objects that each render a pre-generated two triangle plane mesh. Non-procedural tiles can be individually moved, rotated and scaled using their transform components.

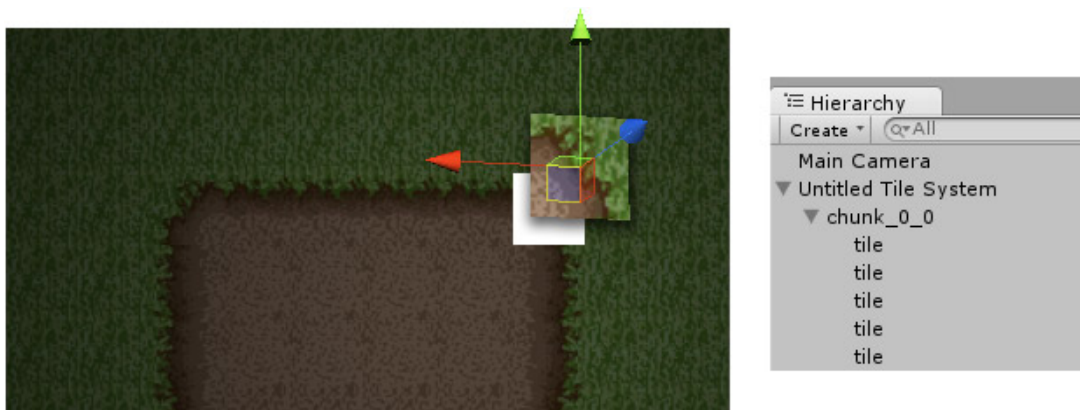


Figure 6-8. Tiles painted using non-procedural brush are individual game objects

It is important to note that non-procedural tiles can be combined by material on a per chunk basis when building a tile system; see [Optimization on page 36](#).

Default Procedural Setting

Tilesets define whether their brushes will be primarily procedural or non-procedural. The value of this setting is usually inherited by brushes, though can be overridden on a per brush basis where necessary.

The value of this property should reflect the primary nature of the tileset because the property can be conveniently overridden on a per-brush basis for special cases (see [Procedural on page 97](#)).



Important

When the value of this property is changed, any tile systems containing tiles that were painted using affected brushes must be refreshed. See [Refreshing all Painted Tiles on page 60](#).

Converting to Procedural

The conversion of brushes from non-procedural to procedural is relatively fast. Existing tile systems should be refreshed to ensure that any tiles that were painted using the affected brushes are updated.

Mesh assets that were previously generated for non-procedural tiles are not removed automatically to avoid breaking tiles that were painted prior to this change. The unwanted mesh assets can be cleaned up, see [Cleanup Unused Meshes on page 94](#).

Converting to Non-Procedural

The conversion process may take a moment to generate missing mesh assets. You may notice that tiles previously painted using the affected brushes have disappeared; this can be corrected by refreshing all affected tile systems.

Working with Tilesets

Tilesets are created using the **Create Brush / Tileset** window which can then be managed using the tileset designer. Tileset brushes can then be created using the tileset designer.

You may find it beneficial to place fully opaque tiles into a separate tileset from those that contain transparency to avoid alpha blending where possible. Alpha blending can be a costly process, especially for mobile platforms.



Tip

Do not create tiles that are fully transparent to produce artificial spaces within oriented brushes. When empty tiles are needed consider using the provided **Empty Variation** master brush instead.

Related information

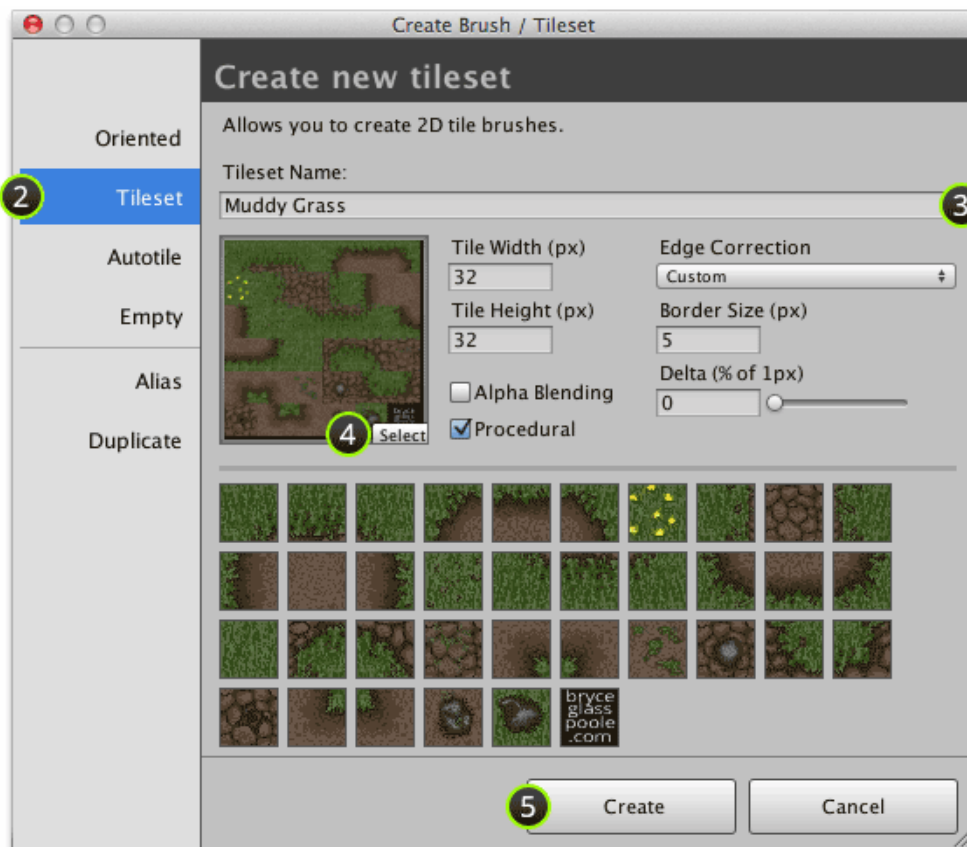
- [Working with Autotile Tilesets on page 101](#)
- [Tileset Designer on page 14](#)
- [Lookup Tileset Brush on page 19](#)

Creating a Tileset

A new tileset can be created by selecting the atlas texture and specifying the tile size and edge correction method. Tilesets are created using the **Tileset** section the **Create Brush / Tileset** window.

Procedure

1. Select menu command  | **Create Brush or Tileset...**



2. Ensure that **Tileset** tab is selected.
3. Input *unique* name for tileset.
4. Select atlas texture and input tile size and [Edge Correction](#) on page 86.
 - **Alpha Blending** - Select when alpha transparency is required to initialize atlas material with the shader **Unlit | Transparent**.
 - **Procedural** - See [Procedural and Non-Procedural Tiles](#) on page 89 to learn more about this parameter.

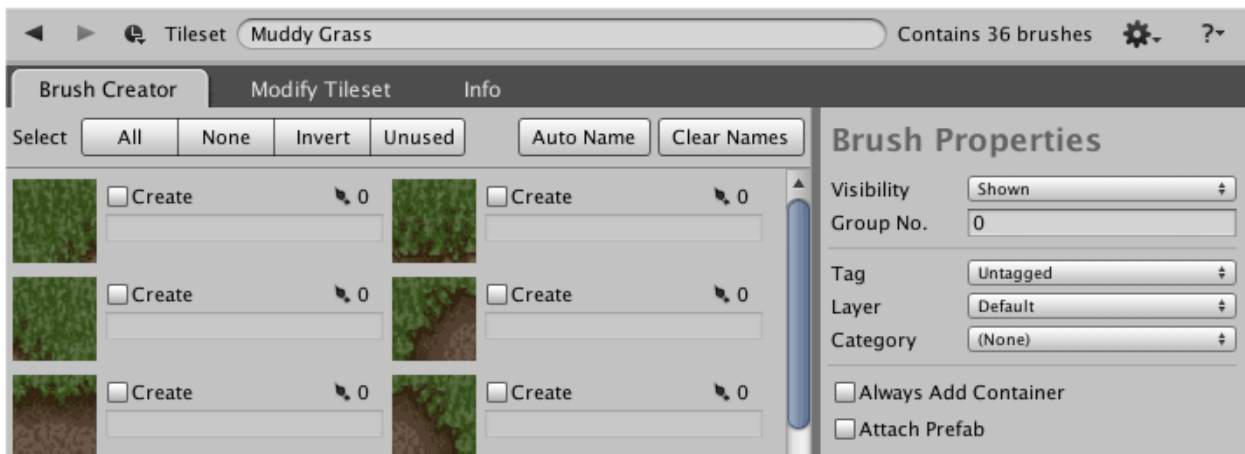
**Note**

Tile previews automatically update when tile size or edge correction parameters are altered. This can help to highlight input errors.

5. Click **Create**.

Results

Tileset should then be shown in designer:



What to do next

You will need to create tileset brushes before you are ready to use your new tileset.

Related information

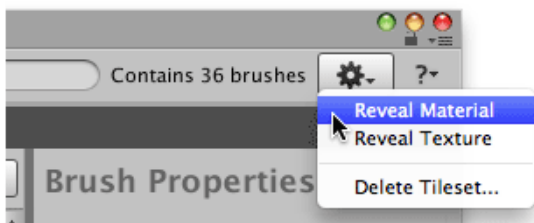
- [Creating Tileset Brushes on page 97](#)

Locating Atlas Material

The atlas material that is automatically created for a tileset can be located easily using the tileset designer. This is useful when you would like to change shader or customize the material.


Locate atlas material from tileset designer:

Select menu command  | **Reveal Material**.

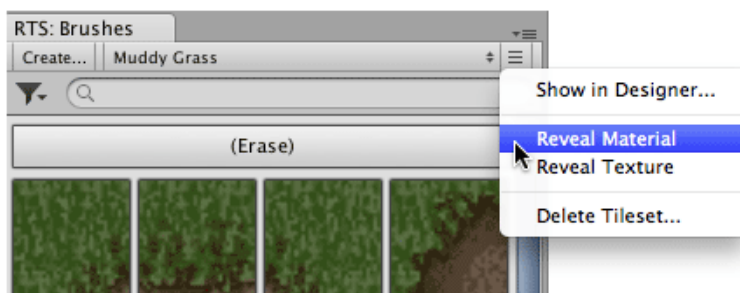


Material asset should then be selected and highlighted in **Project** window

Locate atlas material using brush palette:

Ensure that brush palette is shown ( | **Editor Windows** | **Brushes**).

1. Choose tileset.
2. Select menu command  | **Reveal Material**.



Material asset should then be selected and highlighted in **Project** window

Changing Atlas Texture

Atlas texture of a tileset can be changed using the tileset designer. It is recommended to change the atlas texture using the designer interface since tileset metrics often need to be recalculated.

Procedure

1. Select tileset using **Brush** palette and select menu command ≡ | **Show in Designer....**
2. Select **Modify Tileset** tab.
3. Adjust atlas texture and other properties as needed.
4. Click **Apply Changes**.

Related information

- [Showing Tileset in Designer on page 21](#)
- [Modify Tileset Tab on page 17](#)
- [Default Procedural Setting on page 90](#)

Cleanup Unused Meshes

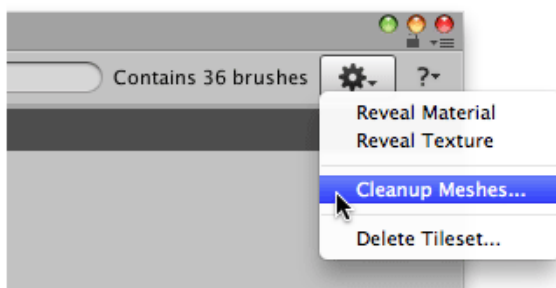
Each non-procedural tileset brush is accompanied by a pre-generated mesh asset. These mesh assets can be removed using the cleanup command when they are not referenced by any tileset brushes.

Before you begin

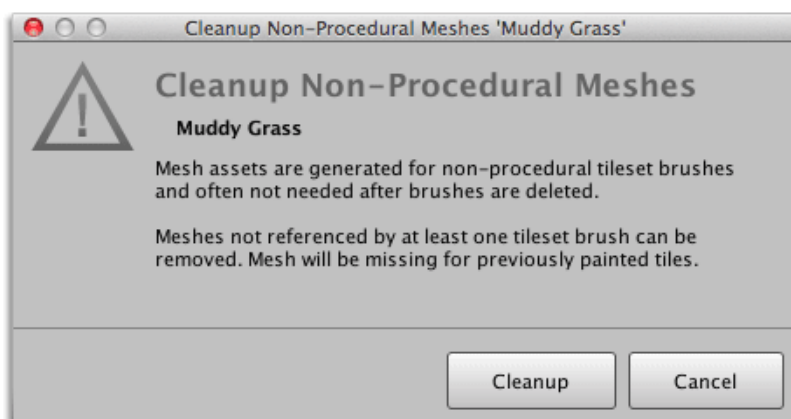
If you have switched tileset brushes from non-procedural to procedural then any tiles that were painted prior will be affected when the 'unused' mesh assets are removed. Such tiles should be refreshed so that they are replaced with procedural tiles instead.

Procedure

1. Select tileset using **Brush** palette and select menu command ≡ | **Show in Designer....**
2. Select menu command ⚙️ | **Cleanup Meshes....**



The following confirmation window should appear:



3. Click **Cleanup**.

Deleting a Tileset

A tileset and all contained brushes can be deleted via the brush palette or alternatively using the tileset designer interface. Tiles that have been painted using a deleted tileset will become damaged.

Options

Related tileset assets can also be removed when deleting a tileset provided that they are stored within the tileset folder. Choosing to delete these assets will affect the rendering of previously painted tiles. Retaining these assets will not preserve procedural tiles.

Delete associated atlas texture

This option is shown when texture asset is located within same folder as tileset asset.

Delete associated material

This option is shown when material asset is located within same folder as tileset asset.

Delete non-procedural mesh assets

This option is shown when pre-generated meshes were generated to support the rendering of non-procedural tiles. Previously painted non-procedural tiles will seem to disappear though their game objects will still exist.



Tip

These assets can be deleted manually at a later time by navigating to the tileset folder in your project.

Before you begin




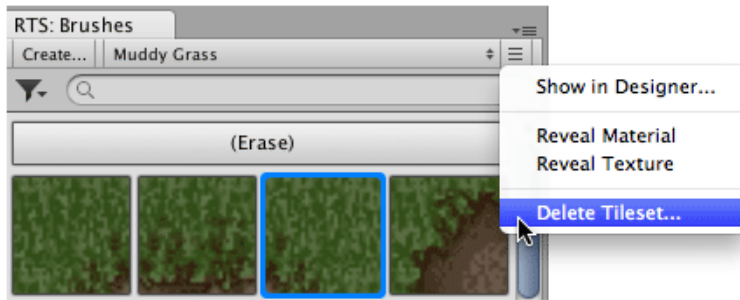
CAUTION

Deleting a tileset will cause all contained brushes to be deleted also. Previously painted tiles will also be affected. This procedure cannot be reversed. This might be a good time to backup your project.

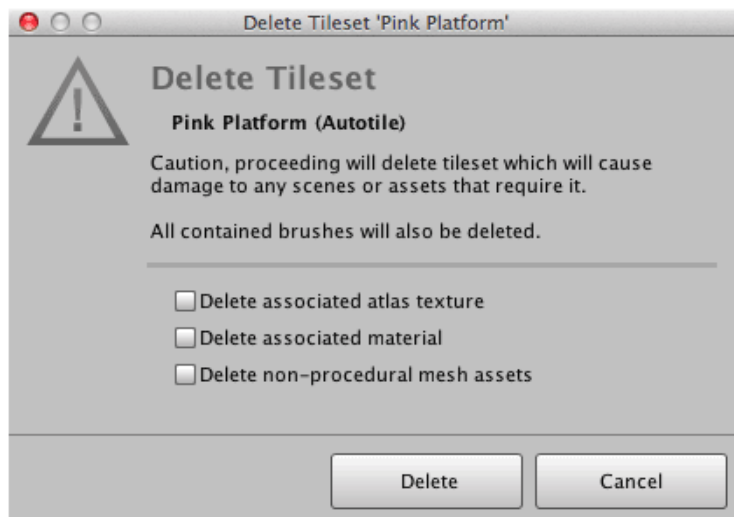
Procedure

1. Select tileset using **Brush** palette.

2. Select menu command  | **Delete Tileset....**



The delete tileset confirmation window will then appear which will look something like the following:



3. Select options as desired.
4. Click **Delete** to delete tileset and all contained brushes.

What to do next

Tiles previously painted using the deleted tileset will no longer reference the brushes that were used to paint them. Non-procedural tiles may appear unaffected depending upon which options you selected when deleting the tileset. Procedural tiles will be affected the next time they are regenerated.

You should repair tile systems to remove any tiles that were painted using a brush that no longer exists.

Related information

- [Repairing a Tile System on page 34](#)

Tileset Brushes

Two-dimensional tiles are painted from tilesets using tileset brushes. Each tileset brush is able to paint a specific tile from the associated tileset.

Related information

- [Designing Brushes on page 61](#)
- [Hiding a Brush on page 82](#)
- [Duplicating a Brush on page 82](#)

- [Deleting a Brush on page 84](#)

Tileset Brush Properties

Tileset brushes have special properties which allow you to attach game objects and define whether they are procedural.

Related information

- [Common Brush Properties on page 62](#)
- [Extended Properties on page 63](#)

Procedural

Tileset brushes will typically inherit the procedural property of their tilesets. This property allows you to mark brushes as procedural or non-procedural on a per brush basis.

This is useful for scenarios where the majority of a tileset is to be used procedurally with the exception of certain select tiles. It is also possible to create both a procedural and non-procedural brush for the same tile.

- **Inherit** - Use to inherit value of [Default Procedural Setting on page 90](#) property from tileset.
- **Yes** - Mark as procedural.
- **No** - Mark as non-procedural.

When this property is changed any tiles previously painted using this brush must be refreshed before changes become apparent. Additional mesh assets will be generated when needed for non-procedural brushes.

See [Procedural and Non-Procedural Tiles on page 89](#) for further information.

Attach Prefab

Additional game objects can be attached to painted tiles by specifying a prefab to attach. This allows you to add components and scripts to each painted tile.

Add Collider

When selected, box colliders are automatically added to each painted tile. Avoid adding colliders unless absolutely necessary because excessive numbers of colliders will incur greater overhead.

This property is only able to add box colliders to tiles. Other types of colliders can be specified by attaching a prefab instead.

Creating Tileset Brushes

Multiple brushes can be created in bulk by selecting and naming each of the required tiles. The same selection of properties will be applied to each created brush.

Procedure

1. Select tileset using **Brush** palette and select menu command  | **Show in Designer...**

2. Select and name tileset brushes as desired.

For each desired tile:

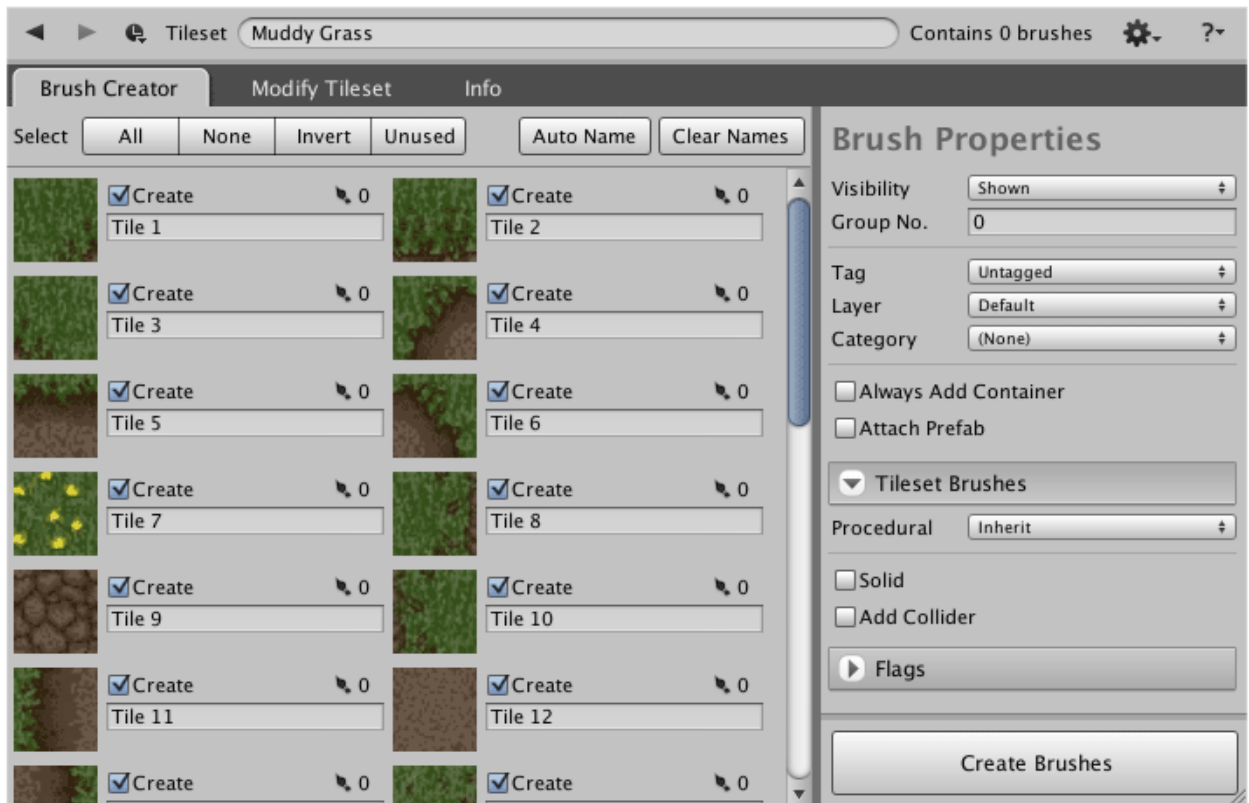
1. Select **Create** tick box.
2. Input unique name for brush.



Tip

Multiple tiles can be selected and named using the utility buttons at the top of the brush creator interface.

You should then have something like the following:



3. (Optional) Adjust [default properties on page 17](#) for brush.
4. Click **Create Brushes** button.

What to do next

The properties of each brush that was created can now be individually customized as needed.

Related information

- [Brush Creator Tab on page 15](#)

Autotile Tilesets

When designing tiles for things like walls, paths and platforms the artist will often need to produce edge pieces and corner pieces. The design process becomes inherently more complicated when inner joins are needed due to the quantity of permutations that are introduced. Autotiles provide an easier way for artists to create such tilesets.



Autotile Artwork

The artwork for an autotile comprises of several key tiles that are each divided into 4 sub-tiles. These sub-tiles can then be mixed and matched to form the atlas of tiles needed for an autotile brush.

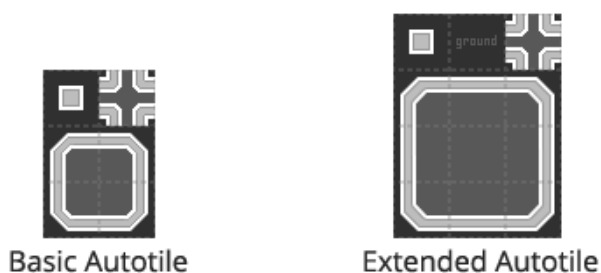


Figure 6-9. Supported autotile layouts

You can design the artwork for an autotile using a regular graphics package. This artwork can then be selected when you create the autotile tileset. An atlas texture will be generated automatically from your autotile artwork upon creating the tileset. The generated atlas is saved as a PNG file which can be modified using a graphics package if desired.



Remember

The atlas of an autotile must be regenerated when changes are made to the input artwork before any modifications are realized.

When designing the artwork for an autotile it is important to keep detail within its respective sub-tile. The "Bad" tile in the following example demonstrates an "upper left" corner tile where the flower overlaps two sub-tiles. This will cause a number of tiles to appear broken like the end piece that is demonstrated beneath.

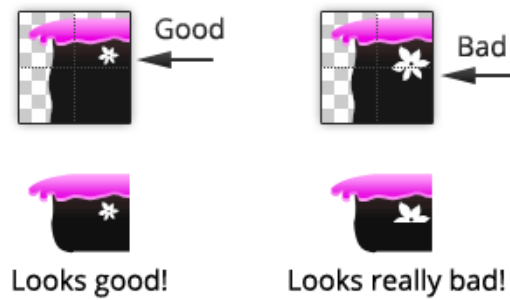


Figure 6-10. Example of good and bad tiles in autotile artwork



Tip

The larger white flower could be added to the "upper left" corner tile by modifying the generated atlas texture instead of placing it within the autotile artwork. Any manual modifications will be lost if autotile atlas is regenerated.

Basic Layout

The basic autotile layout has 6 key tiles each subdivided into 4 smaller tiles that can then be automatically mixed and matched to form 46 tiles.



Remember

Tiles can be of any even size. Borders can be automatically added to generated tiles but must not be added manually to the input artwork.

Extended Layout

The extended autotile layout has dedicated corner and edge tiles allowing you to add greater detail and define rounder edges. The layout consists of 12 key tiles that subdivided into 4 smaller ones to form an output atlas of 47 tiles.



Remember

Tiles can be of any even size. Borders can be automatically added to generated tiles but must not be added manually to the input artwork.

Autotile Expansion

Tile artwork for each available orientation is automatically composed from the input autotile artwork. Borders can be automatically added around each tile to counteract the effects of edge bleeding.

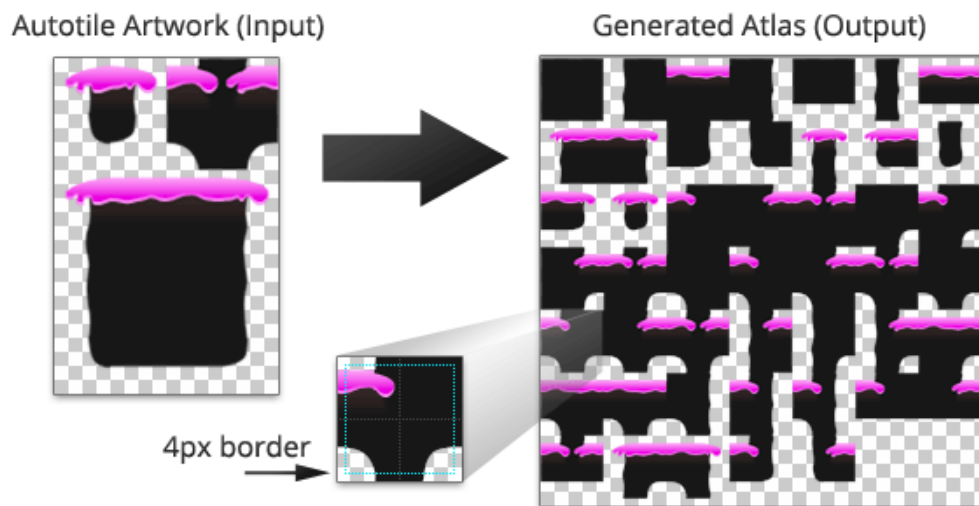


Figure 6-11. Expansion of basic autotile artwork into atlas

Working with Autotile Tilesets

Autotile tilesets are created using the **Create Brush / Tilesset** window which can be managed using the tileset designer. Both autotile and regular tileset brushes can then be created from the tileset using the brush creator interface.

Related information

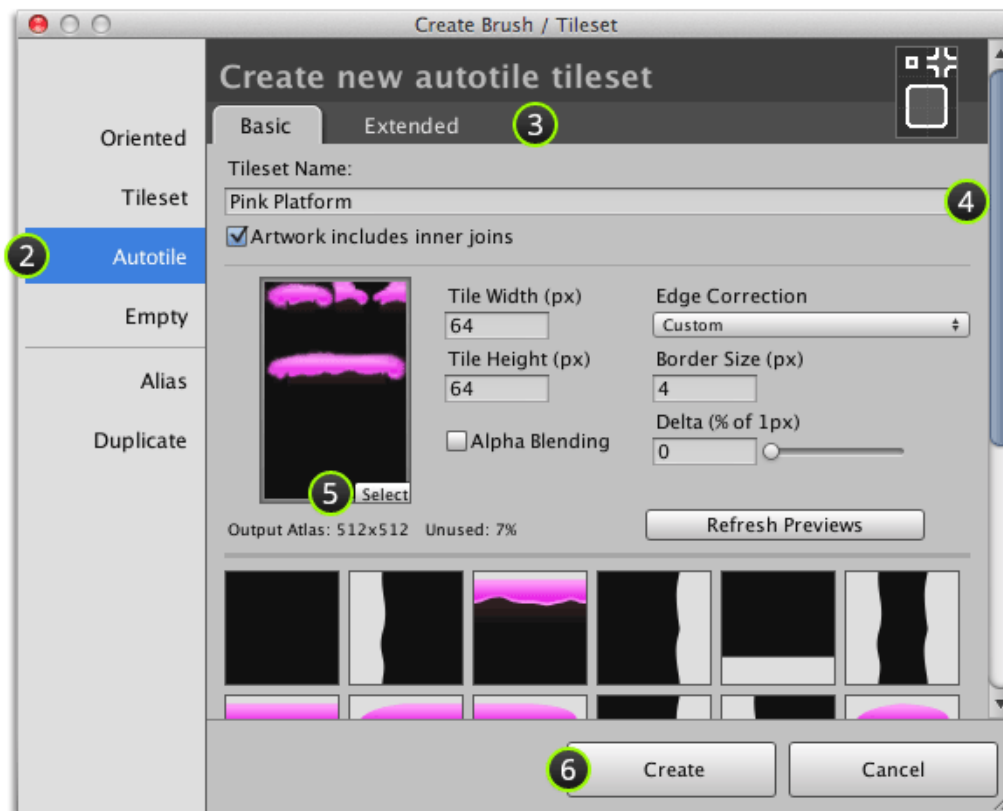
- [Working with Tilesets on page 91](#)

Creating an Autotile Tileset


A new tileset can be created by providing the artwork for an autotile and specifying the tile size and edge correction method using **Autotile** section of **Create Brush / Tilesset** window.

Procedure

1. Select menu command  | **Create Brush or Tilesset...**



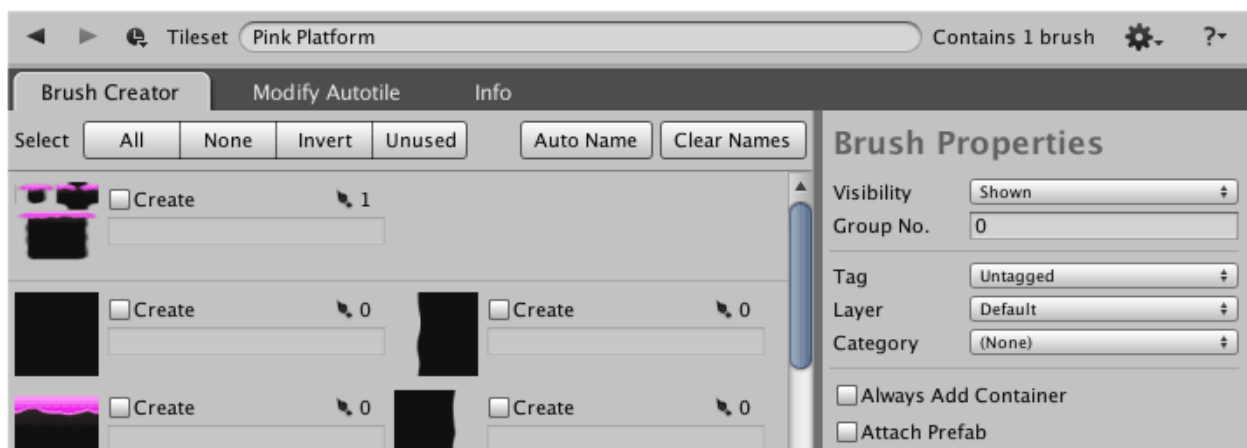
2. Select **Autotile** section.
3. Select tab for the type of autotile artwork that you have.
4. Input *unique* name for tileset.
5. Select autotile artwork and input tile size and [Edge Correction](#) on page 86.
 - **Alpha Blending** - Select when alpha transparency is required to initialize atlas material with the shader **Unlit | Transparent**.

 **Note**
Borders can be automatically added for autotile brushes and is usually the better choice. When specifying border size take care to reduce the area of unused atlas space whilst keeping atlas as small as possible.

6. Click **Create**.

Results

Tileset should then be opened in designer window:



What to do next

You will need to create autotile brushes and/or tileset brushes before you are ready to use your new tileset.

Related information

- [Creating an Autotile Brush on page 105](#)


Regenerate from Autotile Artwork

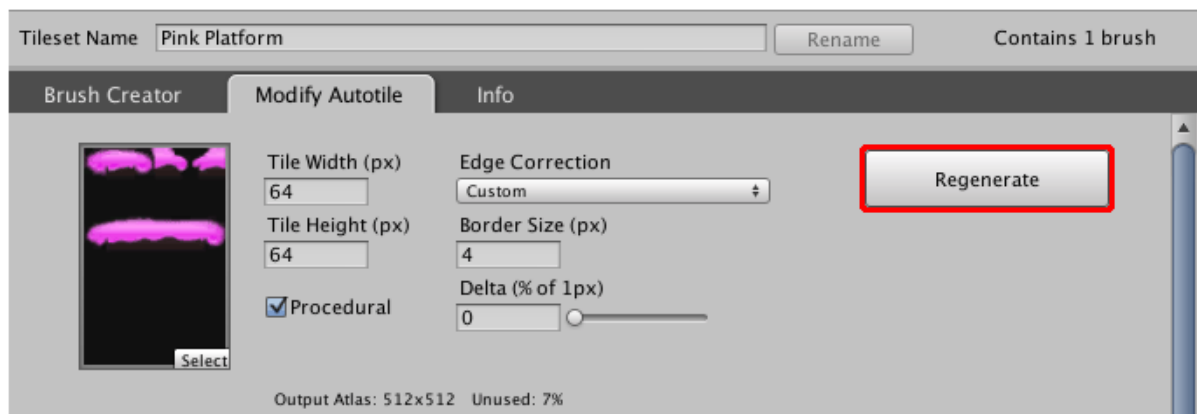
An autotile atlas can be regenerated to reflect changes that have been made to the input artwork. Previously painted tiles will be updated automatically to reflect such changes.

About this task

The generated atlas texture will be overwritten if autotile atlas is regenerated. You should retain a backup of any modifications that you have made to the atlas texture just in case you accidentally lose them.

Procedure

1. Select tileset using **Brush** palette and select menu command  | **Show in Designer....**
2. Select **Modify Autotile** tab.



3. Click **Regenerate**.

Related information

- [Showing Tileset in Designer on page 21](#)
- [Modify Tileset Tab on page 17](#)
- [Default Procedural Setting on page 90](#)

Modifying an Autotile Tileset

The artwork and edge correction properties of an autotile can be modified using the tileset designer. Changes will be applied to previously painted tiles.

About this task

The generated atlas texture will be overwritten if autotile atlas is regenerated. You should retain a backup of any modifications that you have made to the atlas texture just in case you accidentally lose them.

Procedure

1. Select tileset using **Brush** palette and select menu command \equiv | **Show in Designer....**
2. Select **Modify Autotile** tab.
3. Adjust autotile artwork and other properties as needed.
4. Click **Apply Changes**.

Autotile Brushes

Autotile brushes are effectively a subset of oriented brushes which automatically pick tiles based upon their orientation with other tiles. Autotile brushes are only able to paint procedural tiles.



Tip

It is useful to note that autotile brushes can orientate against other autotile or orientated brushes when using the [coalescing properties](#) on page 69.

Related information

- [Designing Brushes](#) on page 61
- [Hiding a Brush](#) on page 82
- [Duplicating a Brush](#) on page 82
- [Deleting a Brush](#) on page 84

Autotile Brush Properties

Colliders are often wasted on inner tiles that cannot be reached. Autotile brushes allow you to specify whether a collider is added for inner and/or outer tiles allowing you to avoid adding unnecessary colliders.

Outer tiles cover the perimeter of an area of tiles; inner tiles are those that are entirely surrounded by other related tiles:

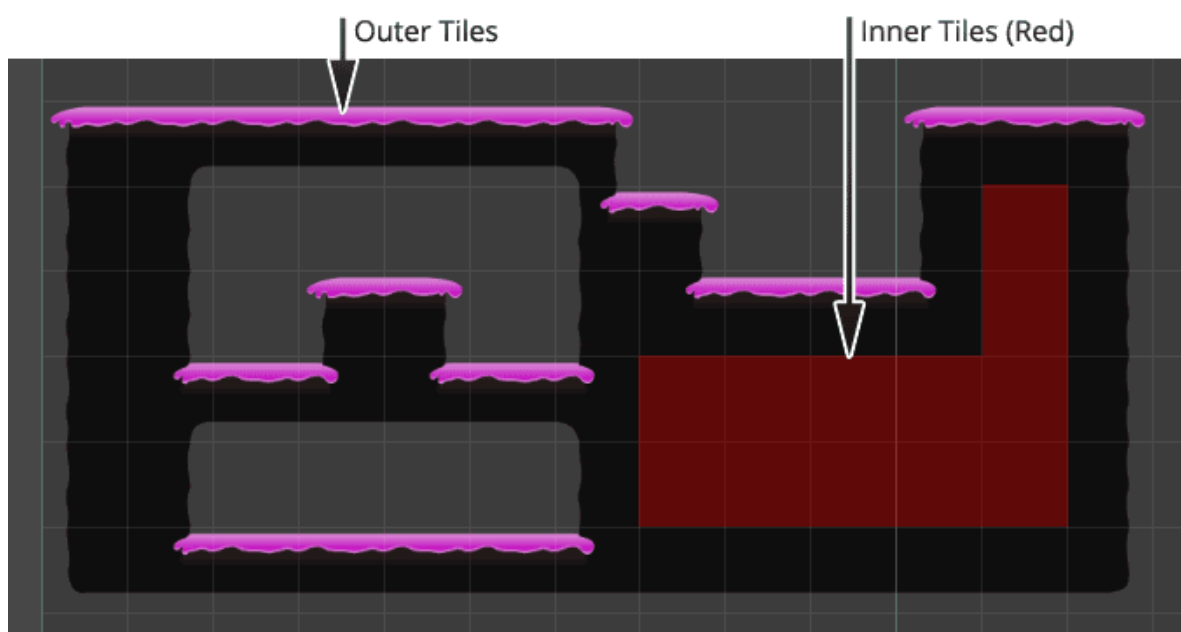


Figure 6-12. Illustration of inner and outer tiles

**Tip**

Colliders come at a cost and should only be added when absolutely needed. The solid flag will typically not incur any overhead, though this will depend upon how you have chosen to utilize it.


Related information

- [Common Brush Properties on page 62](#)
- [Extended Properties on page 63](#)
- [Tileset Brush Properties on page 97](#)

Creating an Autotile Brush

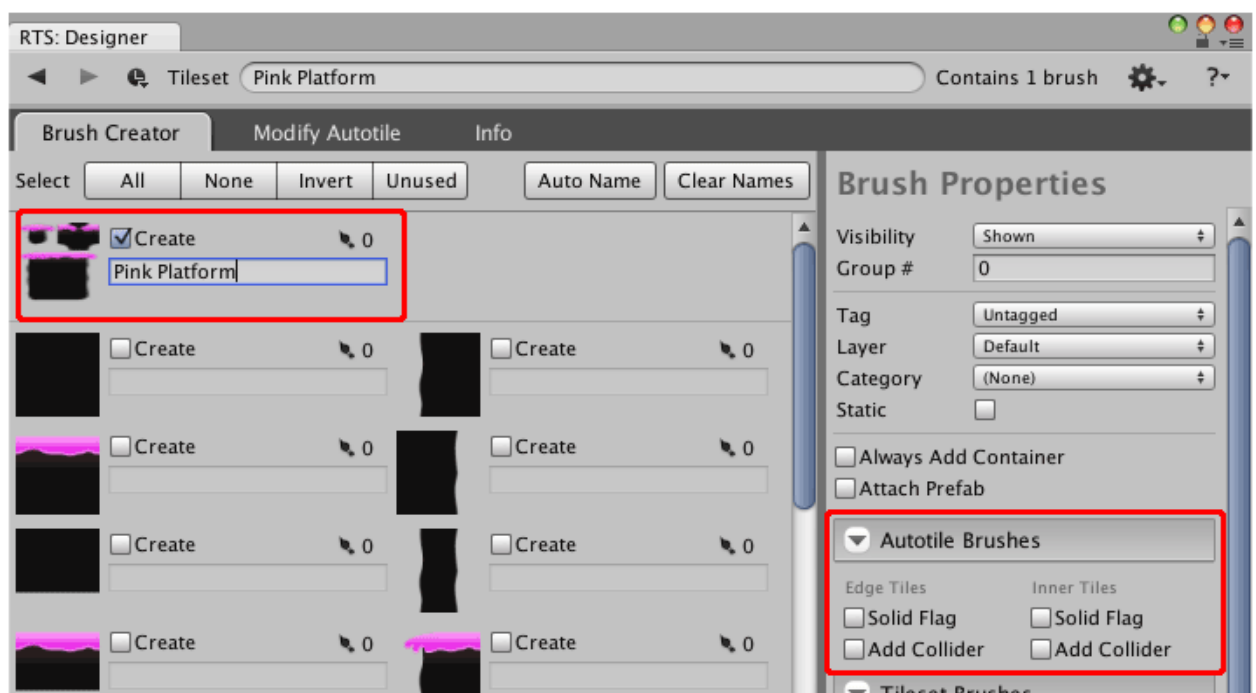
Autotile brushes can be created using the brush creator interface of the tileset designer.

Procedure

1. Select tileset using **Brush** palette and select menu command  | **Show in Designer....**
2. Select and name autotile brush as desired.

Additional options will appear in the **Brush Defaults** panel allowing you to specify autotile-specific details.

You should then have something like the following:



3. (Optional) Adjust [default properties on page 17](#) for brush.
4. Click **Create Brushes**.

Related information

- [Creating Tileset Brushes on page 97](#)
- [Creating an Autotile Tileset on page 101](#)

Chapter 7.

Runtime Scripting

The runtime API allows you to access, paint and erase tiles at runtime using brushes. This can be useful when developing custom tile-based character controllers, level designers, etc.

Here are some of the capabilities of the runtime API:

- Paint and erase tiles using brushes.
- Calculate tile index using ray tracing.
- Find tile nearest given world position.
- Access tile data by row and column indices.
- Use tile data to assist with game logic (orientation, variation index, flags).
- Postpone and control generation of procedural meshes.

For further information please take a look at the [API reference](#).

Consequences of Stripping

Certain runtime functionality becomes unavailable when stripping is applied to a tile system. The level of capabilities lost depends upon the level of stripping that has been applied.

[Stripping Presets on page 37](#) were added to help make the consequences of the stripping process easier to understand. If you attempt to access tile data when tile data has been stripped, you will experience a number of errors including `NullReferenceException` exceptions.



Important

Ensure that tile system component is not stripped when some runtime aspects are required because it is always required when using runtime API. See [Stripping on page 37](#) for more information.

Creating a Tile System Dynamically

Tile systems can be created and used at runtime. This section demonstrates how to implement this.

A tile system is essentially a game object with the `TileSystem` component attached. Once initialized these can be used in the same way as usual. The following source listing demonstrates how to create a tile system at runtime.

C#

```

using UnityEngine;
using Rotorz.Tile;

public class DynamicallyCreateTileSystem : MonoBehaviour {
    // We will use this brush to paint a line of tiles
    public Brush testBrush;

    void Start() {
        // Create game object for tile system
        var systemGO = new GameObject("Dynamic Tiles");
        // Attach tile system component
        var system = systemGO.AddComponent<TileSystem>();

        // Initialize tile system with 10 rows, 15 columns
        // where each tile consumes 1x1x1 in world space
        system.CreateSystem(1f, 1f, 1f, 10, 15);

        // Paint a horizontal line of tiles?
        if (testBrush != null)
            for (int i = 0; i < 15; ++i)
                testBrush.Paint(system, 5, i);
    }
}

```

UnityScript

```

// Assets/DynamicallyCreateTileSystem.js
#pragma strict
import Rotorz.Tile;

// We will use this brush to paint a line of tiles
var testBrush:Brush;

function Start() {
    // Create game object for tile system
    var systemGO = new GameObject('Dynamic Tiles');
    // Attach tile system component
    var system = systemGO.AddComponent.<TileSystem>();

    // Initialize tile system with 10 rows, 15 columns
    // where each tile consumes 1x1x1 in world space
    system.CreateSystem(1f, 1f, 1f, 10, 15, 30, 30);

    // Paint a horizontal line of tiles?
    if (testBrush != null)
        for (var i = 0; i < 15; ++i)
            testBrush.Paint(system, 5, i);
}

```

Figure 7-1. Source Listing: Dynamically creating a tile system

To use the above script:

1. Create a script called "DynamicallyCreateTileSystem.cs".
2. Copy source listing from above.
3. Attach script to a game object and specify a brush using the **Inspector** panel.

Painting at Runtime

The runtime API makes it easy to paint and erase tiles at runtime by reusing the brushes that you have already defined.



Important

Ensure that stripping preset of associated tile system is set to **Runtime Painting, No Stripping** or alternatively **Custom** with appropriate selection.

Painting and erasing tiles

The following source code demonstrates how to paint and erase tiles at runtime. When creating an in-game level designer it is likely that you would want to create some sort of user interface for selecting brushes, etc.

C#

```
using UnityEngine;
using Rotorz.Tile;

public class RuntimePaintingExample : MonoBehaviour {
    // Tile system to paint on
    public TileSystem tileSystem;
    // Brush to use with left mouse button
    public Brush brush;

    void Update() {
        // Find mouse position in world space
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        // Nearest point where ray intersects tile system
        TileIndex ti = tileSystem.ClosestTileIndexFromRay(ray);

        if (Input.GetMouseButtonDown(0)) {
            // Paint with left mouse button
            brush.Paint(tileSystem, ti);
            tileSystem.RefreshSurroundingTiles(ti);
        }
        else if (Input.GetMouseButtonDown(1)) {
            // Erase with right mouse button
            tileSystem.EraseTile(ti);
            tileSystem.RefreshSurroundingTiles(ti);
        }
    }
}
```

UnityScript

```
// Assets/RuntimePaintingExample.js
#pragma strict
import Rotorz.Tile;

// Tile system to paint on
var tileSystem:TileSystem;
// Brush to use with left mouse button
var brush:Brush;

function Update() {
    // Find mouse position in world space
    var ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    // Nearest point where ray intersects tile system
    var ti = tileSystem.ClosestTileIndexFromRay(ray);

    if (Input.GetMouseButtonDown(0)) {
        // Paint with left mouse button
        brush.Paint(tileSystem, ti);
        tileSystem.RefreshSurroundingTiles(ti);
    }
    else if (Input.GetMouseButtonDown(1)) {
        // Erase with right mouse button
        tileSystem.EraseTile(ti);
        tileSystem.RefreshSurroundingTiles(ti);
    }
}
```

Figure 7-2. Source Listing: Painting with brushes at runtime

To use the above script:

1. Create a script called "RuntimePaintingExample.cs".
2. Copy source listing from above.
3. Attach script to a game object and specify the following properties using the **Inspector** panel:
 - The **Tile System** that you would like to paint on.
 - **Brush** to use when painting with left mouse button.

**Note**

A slightly more advanced demonstration is provided with Rotorz Tile System.

See Rotorz/Tile System/Demo/Hat Guy/hatguy_level_designer.unity

Improve performance using bulk edit mode

The performance of painting multiple tiles sequentially can be improved using [bulk edit mode](#):

- Tiles that are painted using orienting brushes (including autotile brushes) can change multiple tiles as their surrounding tiles are changes. During bulk edit mode the actual creation of tiles is suppressed until all changes have been made.
- Procedural tiles are presented using procedurally generated meshes which only need to be updated once regardless of how many tiles have been changed.

C#

```

void PaintTwoTiles(int row, int column) {
    tileSystem.BeginBulkEdit();
    brush.Paint(tileSystem, row, column);
    tileSystem.RefreshSurroundingTiles(row, column);
    brush.Paint(tileSystem, row, column + 1);
    tileSystem.RefreshSurroundingTiles(row, column + 1);
    tileSystem.EndBulkEdit();
}

void PaintExampleTiles() {
    tileSystem.BeginBulkEdit();
    PaintTwoTiles(5, 5);
    PaintTwoTiles(6, 5);
    tileSystem.EndBulkEdit();
}

```

UnityScript

```

function PaintTwoTiles(row:int, column:int) {
    tileSystem.BeginBulkEdit();
    brush.Paint(tileSystem, row, column);
    tileSystem.RefreshSurroundingTiles(row, column);
    brush.Paint(tileSystem, row, column + 1);
    tileSystem.RefreshSurroundingTiles(row, column + 1);
    tileSystem.EndBulkEdit();
}

function PaintExampleTiles() {
    tileSystem.BeginBulkEdit();
    PaintTwoTiles(5, 5);
    PaintTwoTiles(6, 5);
    tileSystem.EndBulkEdit();
}

```

*Figure 7-3. Source Listing: Bulk editing tiles at runtime***Improve performance with pooling**

By default tiles are painted using the [default runtime object factory](#) which simply instantiates tile prefabs to create tiles, and destroys tiles upon erasing them. This is generally acceptable for in-game level designers or when generating tiles at the start of a level, though you should consider a pooling solution if painting and erasing tiles in-game.

A pooling system can be utilized by providing a [custom object factory](#) implementation. You might opt to implement your very own specialized pooling solution, or simply to integrate a third-party solution.

An open-source object factory called [RtsPoolManagerObjectFactory](#) has already been implemented which provides support for [PoolManager](#) by Path-o-logical Games.

**Tip**

A pooling solution is only beneficial when using tile prefabs or prefab attachments. Tiles painted from a procedural tileset would otherwise not benefit from pooling.

Accessing Tiles

Each non-empty tile is represented by a data object which describes its state and references the attached game object when applicable. Tiles can be accessed via the tile system component or on a per-chunk basis.

Accessing a single tile

Individual tiles can be accessed simply by specifying the row and column of the tile that you would like to lookup:

C#

```
TileData tile = tileSystem.GetTile(4, 5);
if (tile != null) {
    // Tile exists!
}
```

UnityScript

```
var tile = tileSystem.GetTile(4, 5);
if (tile != null) {
    // Tile exists!
}
```

Figure 7-4. Source Listing: Access tile on row 4 column 5

Enumerating tiles by row and column

Tiles can be enumerated by row and column transparently of the underlying data structure:

C#

```
for (int row = 0; row < tileSystem.RowCount; ++row) {
    for (int column = 0; column < tileSystem.ColumnCount; ++column) {
        TileData tile = tileSystem.GetTile(row, column);

        // Skip empty tile
        if (tile == null)
            continue;

        // Do something with tile!
    }
}
```

UnityScript

```
for (var row = 0; row < tileSystem.RowCount; ++row) {  
    for (var column = 0; column < tileSystem.ColumnCount; ++column) {  
        var tile = tileSystem.GetTile(row, column);  
  
        // Skip empty tile  
        if (tile == null)  
            continue;  
  
        // Do something with tile!  
    }  
}
```

Figure 7-5. Source Listing: Loop through all tiles in tile system

Enumerating tiles by chunk (more efficient)

Tiles are stored in a chunked data structure which helps to avoid wasting memory when large areas of a tile system are empty. This trait can also be useful when enumerating all non-empty tiles because you can easily skip empty chunks when enumerating them.

**Note**

Row and column indices are not known when enumerating tiles in this manner.

C#

```
foreach (Chunk chunk in tileSystem.Chunks) {  
    // Skip missing chunk!  
    if (chunk == null)  
        continue;  
  
    foreach (TileData tile in chunk.tiles) {  
        // Skip empty tiles in chunk. When accessing tiles  
        // manually we must also check to see if non-null  
        // tile is empty.  
        if (tile == null || tile.Empty)  
            continue;  
  
        // Do something with tile!  
    }  
}
```

UnityScript

```
for (var chunk in tileSystem.Chunks) {  
    // Skip missing chunk!  
    if (chunk == null)  
        continue;  
  
    for (var tile in chunk.tiles) {  
        // Skip empty tiles in chunk. When accessing tiles  
        // manually we must also check to see if non-null  
        // tile is empty.  
        if (tile == null || tile.Empty)  
            continue;  
  
        // Do something with tile!  
    }  
}
```

Figure 7-6. Source Listing: Loop through non-empty tiles in tile system

Chapter 8.

Editor Scripting

The editor API allows you to extend and alter the functionality by creating your own tools, brush creation interfaces, brush designer interfaces and more!

This section of the user guide provides some basic information regarding custom editor scripting. Take a look at the [API editor reference](#) for further information.

Custom Tools

You can implement your very own tools for interacting with tile systems which are shown alongside the provided tools in the **Tools** palette. For consistency we recommend that custom tools should only interact with the active tile system.

A custom tool can be defined by creating a new class extending `ToolBase` which can then be registered using `ToolManager.RegisterTool`. Refer to previous link for simple example implementation of a custom tool.

To improve the usability of your tool you should design an appropriate 22x22 icon graphic. You might decide to create two versions of your icon to suite both the pro and non-pro skins that are provided by Unity if you intend to distribute your custom tool.

Chapter 9.

Frequently Asked Questions

Is this product compatible with iOS and Android?

Yes, absolutely. Munchy Bunny! (available from app store) does in fact use this system.

Of course, the usual mobile development rules still apply, however. Where possible use the built version of your scene when deploying to mobile devices and keep the number of unique materials to a minimum to keep the number of draw calls as low as possible. Where possible use texture atlases (combine textures).

Can pooling be used to improve performance at runtime?

A custom pooling solution can be integrated into the painting/erasing of tiles (or attachments) that are instantiated from prefabs by providing your own [object factory](#) implementation. Effectively this allows you to integrate an existing solution or to implement your very own.

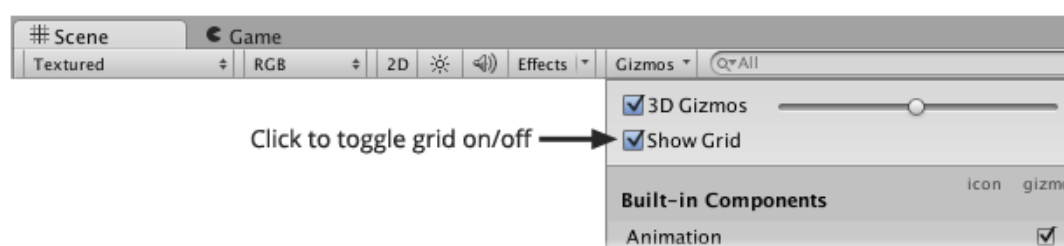
An object factory adding support for [PoolManager by Path-o-logical Games](#) is already available in the form of the open-source script [RtsPoolManagerObjectFactory](#).

Can I interact with tile systems using PlayMaker?

A selection of PlayMaker actions are available from the following open-source repository [RtsPlayMakerActions](#).

Can I hide the default grid?

The grid shown in scene views can cause confusion when working with tile systems. Fortunately this can be hidden via the **Gizmos** drop-down menu in the scene view:



Can I hide colliders when painting?

Yes along with any other gizmos that get in the way of your creativity. Deselect **BoxCollider** (along with any other undesirable gizmos) from the **Gizmos** drop-down menu.

How are the class libraries (DLL) used when building my project?


This product includes two DLL modules. One of these provides functionality for the Unity editor whilst the other provides components necessary to maintain a working tile system at runtime.

The editor DLL is not included in builds, but the other DLL is. If there is no requirement to maintain functional tile systems at runtime then ensure that "Strip System Components" is ticked when building your system.

The runtime DLL can be excluded from your project build manually by dragging the library into an "Editor" folder. It is essential that the DLL is relocated using the Unity editor to avoid breaking existing systems.

A number of warning messages will be presented in the editor console window at runtime for scenes that contain runtime components. See [Stripping on page 37](#).

Is there a way to use a single palette window?

Yes. Since version 2.0.0 the editor window was separated into 2 palette windows. You can select **Classic Mode** under using preferences window via  | **Preferences...** to combine the tool and brush palettes.

Chapter 10.

Troubleshooting

Project contains redundant preview assets

Preview assets are automatically generated for brushes so that thumbnail images can be shown within various user interfaces. If you suspect that your project contains redundant preview prefab assets then you can simply delete them using the project window.

Symptoms

Project contains redundant preview assets within `{Project Path}/Assets/TileBrushes/Editor/Preview/`.


Causes

This usually occurs when tile brushes are deleted manually.

Resolving the problem

Redundant preview prefab assets can be safely deleted. Preview prefab assets will be automatically regenerated if deleted by mistake.

An easy way to ensure that your project contains no redundant preview assets is to simply delete the preview folder which will cause preview assets to be regenerated:

1. Delete folder `{Project Path}/Assets/TileBrushes/Editor/Preview`.
2. Select  | **Rescan Brushes**.

Advice

To avoid this problem ensure that brushes and tilesets are deleted using **Brush** palette or **Designer** window.

Unable to paint on tile system

Unable to paint or erase tiles from the active tile system.

Symptoms

When tile system is selected it is not possible to paint tiles.

Causes

Possible causes for this problem:

- Tile system has been locked.
- Tool option **Paint Around Existing Tiles** is selected.
- Tile system is not active or selected.
- Paint tool is not selected.
- Tile system is missing the `TileSystem` component.
- Tile system has been optimized.
- Tile system must be upgraded to work with newer version of Rotorz Tile System.

Resolving the problem

Below are some troubleshooting steps to help determine the cause of the problem.

Make sure that your tile system is selected in **Hierarchy** window or **Scene** palette. Then try to select the **Paint** tool.

Check if tile system has been locked

1. Select root game object of tile system.

If the message `Tile system is locked. Select 'Toggle Lock' from context menu to unlock inspector.` is shown then the tile system has been locked.

Resolution:

- Select **Toggle Lock** from context menu of tile system component inside inspector window.
- or, Right-click tile system using scene palette and select **Lock** from context menu.

Check if Paint Around Existing Tiles is selected

If this tool option is selected then it is not possible to paint over existing tiles.

Check if tile system has been optimized

1. Select root game object of tile system.

If the message `Tile system has been built and can no longer be edited.` is shown then it is no longer possible to edit this tile system.

Make sure that tile system component is present

1. Select root game object of tile system.
2. Check if **Tile System** component is listed in **Inspector**.

If tile system component is not listed then it is likely that the tile system has been optimized and can no longer be edited. Ensure that you are attempting to paint on the original non-optimized tile system.

Make sure that tile system component is not marked as missing

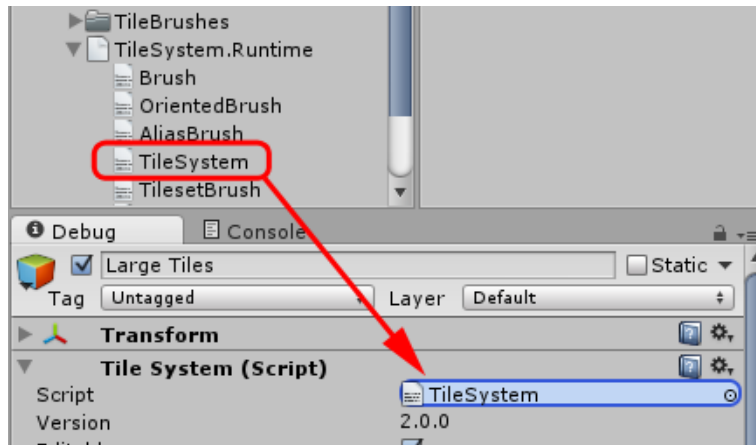
1. Select root game object of tile system.
2. Check tile system component in **Inspector** to verify whether component is valid. Component is not valid if:
 - It is missing the specialized inspector user interface.

- The **Script** property is blank or marked as missing.


If `TileSystem` script is not attached to component, ensure that the path `{Project Path}/Assets/Rotorz/Tile System/TileSystem.Runtime.dll` exists.

Missing DLL can be restored by reimporting the Rotorz Tile System package. You will often need to complete the following steps after restoring missing scripts.

3. Expand `TileSystem.Runtime.dll` inside **Project** window by clicking the arrow next to it.
4. Drag `TileSystem` script from **Project** window and drop onto **Script** property in **Inspector**, like illustrated below:



Upgrade tile system for newer version of Rotorz Tile System

1. Select menu  | **Editor Windows** | **Upgrader**.
2. Review and follow on-screen upgrade steps.

Advice

- Always keep regular backups of your projects.
- Ensure that you are editing the original tile system (as opposed to an optimized version). If you have used the "Build Scene" function ensure that you are editing the original scene.

Changes not reflected when brush is modified

Some brush properties are applied when painting tiles. Changes to such properties will have no immediate effect on previously painted tiles.

Symptoms

Previously painted tiles do not reflect changes that have been made to one or more brushes.

Resolving the problem

Refresh affected tile system(s) to repaint all existing tiles using the updated brush.

Related information

- [Refreshing all Painted Tiles on page 60](#)

- [Refreshing a Single Tile on page 59](#)

Chunk size too large for procedural mesh

Procedural meshes are generated on a per chunk basis but cannot be generated when the maximum number of vertices has been exceeded.

Symptoms

Procedural mesh is not updated and an error message is logged in **Console** window saying `chunk size of 'X' is too large for procedural mesh`.

Causes

There is a hard limit of 64K vertices per mesh which is exceeded when chunk size is too large.

Resolving the problem

Reduce chunk size of affected tile system so that maximum vertex count is not exceeded. The maximum chunk size for tile systems that contain procedural tiles has an area of 100x100 tiles (though not necessarily square).

Related information

- [Changing the Chunk Size on page 27](#)

Build error "count <= std::numeric_limits<UInt16>::max();"

Merged meshes must not exceed the maximum number of vertices per mesh. The chunk size or combine method should be configured in a way that is appropriate for your requirements.

Symptoms

The following error message is logged to the console window when attempting to build one or more tile systems:

```
count <= std::numeric_limits<UInt16>::max();
```

Causes

This error occurs when the number of vertices in the combined mesh exceeds the maximum number of vertices permitted in a mesh (64K).

Resolving the problem

Reduce the size of the combined meshes.

Deselect Combine into submeshes:

When using multiple materials you can sometimes get a little extra mileage before encountering the 64k vertex limit by combining tiles into a separate mesh for each material. Though if you continue to encounter the 64k vertex limit you will need to reduce the size of your chunks.

Reduce chunk size:

The following advice is based upon the selected **Combine Method**.

Combine Method	Advice
By Chunk	Specify a smaller chunk size using the "Custom Chunk in Tiles" combine method instead.
By Tile System	Consider using one of the other combine methods instead.
Custom Chunk In Tiles	Specify smaller chunk size.

You may want to consider the following if the above advice does not help you to resolve your problem:

- Avoid selecting "Static" property for some brushes so that not all tiles are combined.
- Utilize the static batching feature of Unity Pro instead.
- Reduce the number of vertices in your tile meshes.

Actor passes through inner tiles of smooth platform

Actor passes through tiles painted using the smooth platform brush because inner tiles do not have colliders by default.

Symptoms

Collisions are not detected between objects and inner tiles (those that are surrounded by 8 other tiles).

Causes

The smooth platform brush does not include a box collider on inner tiles by default because they are usually not needed for 2.5D platform type games. Omission of these colliders improves performance both in editor and in game.

Resolving the problem

Resolution #1 - Better when possible:

Manually add minimum number of colliders using empty game objects instead of using colliders in brushes. The fewer colliders the better!

Resolution #2 - Less efficient but often easier:

1. Find and select the following prefab in **Project** panel: **Rotorz/Tile System/TileBrushes/Blocks/Smooth Block/centre_inner.prefab**
2. Select menu **Component | Physics | Box Collider**
3. Ensure that centre of collider is set to (0, 0, 0) and that size of collider is set to (1, 1, 1) using **Inspector**.

Tiles not positioned, rotated or scaled as expected

Tiles do not appear to be positioned, rotated or scaled correctly when painted because an offset has been specified or its origin (pivot point) is incorrect.

Symptoms

Painted tiles are not aligned properly against tile system.

Causes

Possible causes for this problem:

- An incorrect transform offset has been specified for brush.
- **Apply Prefab Transform** is selected for brush but transform of tile prefab is incorrect.
- Origin of tile prefab is incorrect.

Diagnosing the problem

Check offset of prefab

Prefab transform is used to offset tile when **Apply Prefab Transform** is selected for brush. The position value of a prefab is often non-zero upon creation which may need to be manually set to (0, 0, 0).

1. Find and select prefab asset using **Project** panel.
2. Check position using **Inspector**.
3. See [Fix offset of prefab on page 122](#) if position is something other than (0, 0, 0).

Check pivot point of prefab

1. Create a blank scene
2. Add instance of troublesome prefab to scene.
3. Ensure that position of prefab instance is set to (0, 0, 0) using **Inspector**.
4. See [Fix pivot point of prefab on page 122](#) if object does not appear at center of scene.

Resolving the problem

Fix offset of prefab


1. Find and select prefab asset using **Project** window.
2. Set position of prefab to (0, 0, 0) using **Inspector** for central placement within tile cell.
3. Attempt to paint using brush to see if this resolves the problem.

Fix pivot point of prefab

Correct pivot point of game object by repositioning nested objects and/or altering mesh if required. Another easier way to fine-tune tile object placement is to apply a prefab offset as described in the next section.

Fix placement by applying prefab offset

1. Create blank scene and add a tile system.

2. Paint instance of the troublesome tile.
3. Select root most object of tile which is connected to tile prefab.
4. Correct position/rotation/scale of tile using the regular Unity tools or inspector.
5. Select  | **Use as Prefab Offset**.
6. Attempt to paint using brush to see if this resolves the problem.

Related information

- [Offset, Rotate and Scale Prefabs on page 65](#)

Editor performs slowly with lots of tiles

Unfortunately your hardware and the Unity software can only handle so much! There are several tips that can help to overcome some of these limitations.

Symptoms

Unity editor begins to perform slowly when scene contains a lot of tiles (amongst other game objects).

Resolving the problem

Tip #1 - Simplify brush prefabs

Ensure that the prefabs associated with brushes contain the bare minimal number of components and colliders because these can have a severe impact upon performance (both in editor and at runtime).

- Remove collider components and manually create and position larger colliders using game objects to reduce overall number of colliders. For example, instead of having 10 box colliders (one per tile) in a line for a platform, use 1.
- Minimize number of script components that are attached to tiles to the bare minimum.

Tip #2 - Use multiple smaller tile systems

Compose scenes using multiple smaller tile systems instead of one large one. This can help to improve performance in editor.

Tip #3 - Use fewer (but larger) tiles

Try to reduce the number of tiles by creating larger tiles with larger (combined) meshes, simpler colliders and fewer overall components. The tile system would thus contain less tiles (whilst being the same physical size) meaning that there are less game objects in the Unity editor. This can give a massive performance boost.

Tip #4 - Compose scene from multiple smaller optimized tile systems

1. Create each tile system in a separate scene with appropriate chunk sizes.
2. Ensure that appropriate stripping and build options are specified. It is important to ensure that static tiles are merged.
3. Build optimized version of each tile system as prefabs.
4. Compose final scene using the optimized prefabs to overcome some of these limitations.

**Important**

Ensure that brushes are marked as "static" in order to benefit from these optimizations.

Advice

If you are experiencing slow-down in the editor due to large quantities of tiles, you will almost certainly benefit at runtime from optimized tile systems!

Visual "glitches" between tiles

Visual anomalies may be encountered when creating tiles or textures. There are a number of causes of this.

Diagnosing the problem

Texture Tiling

Ensure that tiling of texture is seamless using graphics software.

UV Coordinates

Ensure that UV coordinates are accurate enough for smooth tiling. Zoom into vertices and align them to your texture as accurately as possible.

**Tip**

Make use of the snapping functionality of your modelling software. Some modelling packages allow selected vertices to be snapped together on a single axis (using scale in Blender, for example).

Edge Bleeding

There are a number of factors that can contribute to edge bleeding:

- **Texture filtering** - Bilinear and trilinear filtering causes texture pixels to be blended with other adjacent pixels. Edge bleeding occurs when pixels are incorrectly blended across UV seams.
- **Mip-mapping** - Real-time applications make use of a rendering feature called mip-mapping which helps to improve the performance of rendering textured meshes by storing multiple versions of each texture at reduced qualities. Where possible the lower quality versions are used to improve performance. The process of generating reduced texture sizes causes blending to occur across UV seams.
- **Lossy texture compression** - Compression artefacts exist within compressed versions of textures. Whilst the effects of this are often not prominent, they can lead to edge bleeding in atlas textures where such artefacts cross UV seams.
- **Use of Non-Power-Of-Two (NPOT) textures** - It seems that NPOT textures are internally resized into power-of-two textures. The process of upscaling/downscaling the texture causes pixels to be blended and thus leading to artefacts when using atlas textures.

In most scenarios the effect of edge bleeding can be counteracted by adding suitable borders around each UV island in your textures. Refer to [Edge Correction on page 86](#) for further information regarding edge correction with respect to 2D tilesets.

Smooth Brushes and Building

- When appropriate enable brush smoothing. This will smooth normals across tiles (provided that topology of tile meshes allow for this).
- Build tile system to benefit from snapped vertices and any smoothed joins.

Vertex Threshold (When Building)

Ensure that snapping vertex threshold is sufficient for the scale that you are working with. The vertex threshold can be altered using the inspector panel by selecting the tile system. Please note that changes will only become apparent when tile system is rebuilt.

Poor texture alignment in tileset brushes

Atlas tile textures do not align properly when painted onto a tile system. There are a number of causes for such issues, this troubleshooting guide covers the most common reported difficulties.

Symptoms

- Pixels from adjacent tiles bleeding through onto painted tile.
- Visual seam between tiles.
- Artwork of multiple tiles appear on painted tile.
- Fraction of tile artwork appears on painted tile.
- Tile texture appears blurred and stretched.

Causes

Possible causes for this problem:

- Incorrect or inconsistent tile sizes in texture itself.
- Tile texture does not support seamless tiling.
- Side effect called "Edge Bleeding" which has several contributing factors including:
 - Mip-mapping
 - Bilinear / trilinear texture filtering
 - Lossy texture compression
 - Use of Non-Power-Of-Two (NPOT) texture sizes

Diagnosing the problem

Ensure that tile sizes are consistent in atlas texture

Many graphics applications are able to display a grid with user defined cell sizes.

1. Open atlas texture using such software.
2. Enable grid where cell size matches that of your intended tile size.
3. Ensure that each tile fits perfectly within grid.

Where applicable make sure that tiles tile seamlessly

1. Open atlas texture using a graphics package.

2. Copy one of the troublesome tiles and paste into a blank image.
3. Offset tile image by half of its size so that you can see how it joins with itself.
4. If tile doesn't tile against itself then this must be corrected.

Try to disable mip-mapping for atlas texture

If creating an entirely two-dimensional game using an orthographic camera projection then it is unlikely that you will benefit from mip-mapping.

Make sure that texture is square and a power-of-two

1. Load atlas texture into graphics software.
2. Review image dimensions:
If image is not perfectly square or does not have power-of-two dimensions then texture filtering may be the cause of your problem.

Related information

- [Edge Correction on page 86](#)

Transparency not working with tileset brushes

By default tileset brushes use a material with a very simple opaque shader for improved performance. A different shader can be selected when alpha transparency is desired.

Symptoms

Transparency is not working for 2D tileset or autotile brushes.

Causes

Atlas material is using a fully opaque shader.

Resolving the problem

1. Navigate to atlas material (see [Locating Atlas Material on page 93](#)).
2. Select a shader that supports transparency using **Inspector**.
i.e. **Unlit | Transparent**.

Updated tileset not reflected

Changes that have been made to a tileset have not been reflected in previously painted tile systems.

Causes

Possible causes for this problem:

- Non-Procedural tiles need to be refreshed manually.
- Procedural meshes have not updated.

Resolving the problem

- Try to reload the current scene to see if that resolves the issue.
- Manually refresh the affected tile systems.

Related information

- [Refreshing all Painted Tiles on page 60](#)

Brush asset appears to be missing or broken

Brush asset files will be empty if Unity crashes before asset files are saved successfully.

Symptoms

One or more brush asset files appear to be empty with no associated script or data.

Causes

When assets are modified they are usually flagged as being dirty which indicates to Unity that they will need to be saved. If Unity crashes before assets are saved successfully you can end up with empty asset files. This issue is not unique to brush assets.

Advice

Be sure to save assets regularly using **File | Save Project**; doing so will help to avoid this issue.

Glossary

Alias Brush

Brush that makes reference to another non-alias brush allowing some properties to be overridden and materials to be remapped.

Atlas

2D texture that contains artwork for multiple tiles.

Autotile Brush

Autotile brushes require a special tileset which is automatically generated from the input autotile artwork. The aim of autotile brushes is to make it easier for artists to create 2D tiles that automatically orientate like walls, paths, etc.

Brush

Tiles can be painted using specialized brushes. All brush types are derived from the base class `Rotorz.Tile.Brush`. Some brushes are stored as independent asset files whilst others are contained within composite assets like tilesets.

Build Prefab

Optimized version of tile system can be built and then saved as a prefab.

Build Scene

All tile systems in scene can be optimized where the optimized scene is saved to a separate file.

Cell

A tile system is composed of cells which can either be empty or can contain a tile. The content of a cell can be accessed at runtime using `TileSystem.GetTile`.

Chunk

Tile systems and their data structures are broken down into chunks to help reduce memory requirements. Procedural meshes are added on a per chunk basis procedural brushes are used.

Empty Brush

A brush that paints tiles with no visual output. These are useful when designing oriented tiles that require gaps or when defining tiles of a purely logical nature. By default a master brush called "Empty Variation" is provided that can be used within oriented brushes.

Master Brush

Brush that cannot be modified using the brush designer. Master brushes can be copied and aliases can be created using the brush designer. Rotorz Tile System includes some master brushes that can be used to create custom platform tiles simply by providing an alternative material.

Oriented Brush

Brush which automatically picks from a number of user defined orientations and variations to automatically paint tiles that connect with one another.

Orientation

Describes the context of a tile based upon its 8 surrounding tiles. Both oriented brushes and autotile brushes analyse the orientation of tiles to determine which ones should be painted.

Plop

A "plop" is an object which does not contribute to the data structure of a tile system and is 'plopped' using the plop tool. Plops can be erased and cycled in a similar way to regular tiles.

Runtime API

Programming interface that can be used by developers to interact with tile systems at runtime. For example, this could be used to create an in-game level designer.

Tile System

Virtual grid that can contain zero or more tiles.

Tileset

A tileset is associated with an atlas that contains a collection of tiles that can be painted by creating and using tileset brushes. Tilesets provide a small degree of indirection which makes it relatively easy to convert tileset brushes between procedural and non-procedural.

Tileset Brush

Brush that can paint a procedural or non-procedural tile from a tileset.

Tool

Used to paint with brushes and manipulate tiles.

Variation

Multiple tile variations can be specified for each orientation of an oriented brush.

Index

A

Accessing tiles, 111
 Alias Brush, 77
 creating an alias brush, 78
 overriding properties, 77
 Android, 115
 Autotile, 98, 101
 artwork, 99
 basic layout, 100
 extended layout, 100
 autotile brush. *See* Autotile Brush
 creating an autotile tileset, 101
 expansion, 100
 modify autotile tileset, 103
 regenerate from artwork, 103
 Autotile Brush, 104
 brush properties, 104
 creating an autotile brush, 105

B

Brush, 61, 67, 77, 80, 81, 96, 104
 access designer, 20
 alias brush. *See* Alias Brush
 apply prefab transform, 65
 autotile brush. *See* Autotile Brush
 common properties, 13, 62
 brush name, 62
 category, 63. *See also* Brush Category
 favorite, 63
 hide, 63
 layer, 63
 smooth, 37, 63
 static, 62
 tag, 63
 visibility, 63
 creating brushes, 71, 78, 80, 81
 deleting a brush, 84
 designer, 13
 designing brushes, 61
 duplicating a brush, 82
 empty brush. *See* Empty Brush
 extended properties, 13, 63
 always add container, 64
 flags, 65
 force legacy sideways, 64
 group number, 64
 scale mode, 64
 hiding a brush, 82
 master brush. *See* Master Brush
 oriented brush. *See* Oriented Brush
 properties, 77
 remapping materials, 66
 tile offset, 65
 tileset brush. *See* Tileset Brush
 use as prefab offset, 66
 Brush Category, 61, 63

 adding a category, 61
 filtering, 11
 removing a category, 62
 Brush Palette, 9
 filter menu, 11
 keyboard shortcuts, 13
 list presentation, 10
 list views, 10
 search, 12
 Build entire scene, 43
 Build individual tile system, 42
 Build Options, 39

C

Changing size of chunk, 27
 Chunk, 26
 change size, 27
 display boundaries, 22, 26
 Classic Mode, 116
 Class Libraries, 106, 114, 115
 Clearing a tile system, 35
 Coalescing, 69
 Common brush properties, 62
 Creating a tile system, 28, 29. *See also* Presets
 Creating tile systems at runtime, 106
 Cycle Tool, 48

D

Designer Window, 13
 accessing brush designer, 20
 accessing tileset designer, 21
 brush designer, 13
 keyboard shortcuts, 22
 lock, 21
 lookup tileset brush, 19
 selection history, 22
 tileset designer, 14
 DLLs, 106, 114, 115

E

Editor API, 114
 custom tools, 114
 Empty Brush, 80
 creating an empty brush, 80
 Extended brush properties, 63

F

Fill Tool, 49
 enable undo/redo, 22
 limit, 22
 Force Refresh Tiles, 59, 60

G

Getting Started, 1

H

Hiding Colliders, 115
 Hiding Scene View Grid, 115

I

Import Extension Package, 2, 5
 Inspector, 32, 33, 33
 Installation, 2
 iOS, 115

K

Keyboard Shortcuts, 22, 55

L

Line Tool, 49

M

Master Brush, 81
 creating a master brush, 81
 Mobile Compatibility, 115

O

Optimization, 26, 36, 106
 build options, 39
 build tile system, 42, 43
 smoothing joins, 37
 stripping, 37
 vertex snapping, 36, 37
 Orientation Indicator, 55, 56
 Oriented Brush, 67
 coalesce mode, 69
 creating an oriented brush, 71
 debugging, 56
 define orientation, 72
 fallback orientation, 71
 find orientation, 72
 orientation, 67
 add variation, 74
 lookup variation, 75
 remove orientation, 74
 remove variation, 75
 setting default orientation, 73
 smooth property, 63
 variation
 copy, 76
 drag and drop, 76, 76
 randomization, 68
 reorder, 76
 weights, 68

P

Painting, 45
 Painting at runtime, 108
 Paint Tool, 48
 Palette
 brushes, 9
 scene, 8
 tools, 6, 6
 Picker Tool, 49
 PlayMaker, 115
 Plop Tool, 52
 Pooling, 110, 115

Pool Manager, 115
 Presets, 29
 creating a preset, 29
 deleting a preset, 31
 modifying a preset, 30
 Project Workflow, 3

R

Rectangle Tool, 50
 Refresh Tiles, 60
 Repairing a tile system, 34
 Replace by Brush, 58
 Runtime API, 106
 accessing tiles, 111
 consequences of stripping, 106
 creating tile systems, 106
 enumerating tiles, 111
 painting, 108
 runtime options, 41
 Runtime Options, 41

S

Scene Palette, 8
 lock, 9
 rename, 9
 reorder, 9
 sorting, 9
 unlock, 9
 visibility, 8
 Smoothing Joins, 37
 Spray Tool, 51
 Status Panel, 55
 Stripping, 26, 37, 42, 43, 106
 stripping options, 38
 stripping presets, 37

T

Tile Data, 37
 Tile Index, 55
 Tileset, 86, 91, 97
 access designer, 21
 autotile. *See* Autotile
 autotile brush. *See* Autotile Brush
 bleeding, 86
 changing atlas texture, 94
 changing to non-procedural, 97
 changing to procedural, 97
 cleanup unused meshes, 94
 converting to non-procedural, 91
 converting to procedural, 91
 creating a tileset, 91
 deleting a tileset, 95
 designer, 14
 brush creator tab, 15
 info tab, 18
 lookup brush, 19
 modify tileset tab, 17
 edge correction, 86
 border, 88

- none, 87
- uv delta, 89
- flickering artifacts, 86
- locate material, 93
- non-procedural, 89, 90
- procedural, 89, 89
- procedural property, 90
- tileset brush. *See* Tileset Brush
- Tileset Brush, 96
 - brush properties, 97
 - add collider, 97
 - attach prefab, 97
 - procedural, 97
 - creating multiple tileset brushes, 97
- Tile System, 26
 - building, 42, 43
 - change cell size, 33
 - chunks. *See* Chunk
 - clear tiles, 35
 - gizmo colors, 22
 - grid lines, 22
 - immediate previews, 22
 - inspector, 32, 33, 33
 - optimization. *See* Optimization
 - refresh tiles, 59, 60
 - repair broken tiles, 34
 - runtime options, 41
 - stripping. *See* Stripping
 - switching between systems, 55
 - transform, 32
 - visibility, 8
- Tile System Panel
 - classic mode, 22
 - tool selection, 22
- Tool Menu, 6, 6
- Tool Palette, 6
- Tools, 45
 - common options, 45
 - nozzle, 47
 - nozzle radius, 47
 - nozzle size, 47
 - paint around existing tiles, 46
 - randomize variation, 46
 - rotation selector, 45
 - shift variation, 46
 - cycle tool, 48
 - fill tool, 49
 - fill rate, 51
 - line tool, 49
 - paint tool, 48
 - picker tool, 49
 - plop tool, 52
 - alignment, 53
 - alignment axis, 53
 - free placement, 53
 - grid snapping, 53
 - snap, 53
 - snap grid, 53
 - x-alignment, 53
 - y-alignment, 53

- rectangle tool, 50
 - fill, 50
 - outline, 50
- spray tool, 51
- Tools Palette
 - context menu, 8
 - tool buttons, 6
 - tool menu, 6
- Tweaking Tiles, 58

U

- Update Extension, 5
- User Interface, 6
 - inspector, 32, 33, 33
- User Preferences, 22

V

- Variation Index, 55
- Vertex Snapping, 36, 37
- Visibility, 8
- Visualizing Chunks, 26