MeshBrush Documentation

Index

1. Release notes

V1.1

- Fixed a NullReferenceException error when deselecting the GameObject on top of which you were painting meshes.
- Added a "Combine painted meshes" button in the inspector to merge all painted meshes for further performance optimization.
- Added a "Delay" parameter in the inspector to allow painting meshes continuously while holding down the paint button.
- Added the possibility to customize your keyboard shortcuts (per script instance) via the "Customize Keyboard Shortcuts" foldout menu in the inspector.

V1.15

- Added important slope influence values for more control over the painted meshes rotation.
- Added two toggles: one to optionally flip the Y-Axis on painted meshes; one to either keep your painted meshes upright/tangent to their underlying surface.

V1.2

- Generally improved the editor's performance.
- Slope management gets its own group foldout in the inspector.
- New group string in the inspector for visual help to the user.
- Slope filter added.
- Slope reference vector sampling mode added.
- Updated documentation.
- Runtime API script + example script.

V1.3

- You can now delete painted meshes inside the circle brush. (Default deletion key is L).
- Holder object grouping functionality.
- Drastically improved the editor’s performance thanks to the new paint buffer (applies to the deletion too).
- Added option to randomize the number of meshes to paint.
- Added a save functionality to store combined meshes for later re-usage (e.g. via prefabs).
- Updated documentation again (got rid of the ugly .txt file).
- Fixed numerous bugs and stability issues.

V1.4

- Fixed various small bugs.
- Fixed compiler errors on Android and Web Player.
- Prefab connections are now maintained when painting.
- Added the precision placement mode.
- Added the area combine functionality (it's now possible to combine painted meshes only within the brush area).
- New documentation (getting better and better...).

V1.5

- Reorganized UI
    - The precision placement mode is now inside the set of meshes to paint foldout, which makes more sense.
    - New foldout for the brush settings section
    - The huge, plain and raw help text in the inspector got subdivided into separate groups (per feature/functionality)
    - MeshBrush now has its own menu item folder under GameObject/MeshBrush
- Complete remake of the Set of meshes to paint interface (now with pleasant look + useful list functions, instead of a simple default array).
- Added MeshBrush Templates (saving/loading functionality).
- Added a favourites list for favourite templates.
- Added global painting mode.
    - Layer based painting
- Updated docs

V1.6

- Fixed a rare object leak in the paint buffer where saving and restarting the editor in the right moment would not entirely clear the paint buffer (thus leaving some buffered objects in the scene).
- Fixed the hard coded paths for the template favourites list file and some UI icons: it is now possible to move the root MeshBrush folder freely around inside the project and sub-folders.
- Definitively fixed the prefab connection problem: those connections are now maintained in all cases and scenarios where you'd paint prefabs instead of raw meshes.
- Added the new Overlap Filter.

V1.7

- More (small) bugfixes and simplified code.
- MeshBrush templates management and data structure simplified (now much more organized and easier to maintain).
- Added many (necessary) undo stack entries for various MeshBrush related actions (loading up a template can now be undone for instance, as well as deleting fields inside the set of meshes to paint, resetting all randomizers, big changes in various settings and many more…).
- Added a completely reworked and new Modern UI with dynamically resizable preview icons of the various prefabs inside the set of meshes to paint.
    - Drag 'n' drop functionality in both the set of meshes to paint and the favourites list.
    - Switching over seamlessly to the Classic UI is possible: there is a button in the inspector that allows to switch back and forth between the UIs.
- Added the Constant mesh density toggle in the inspector.
    - Use an optional mesh density value to define the amount of meshes to paint instead of a traditional fixed number. This keeps the density of the painted objects homogeneous, even when painting with different brush sizes.

2. What is MeshBrush and how do I get started?

MeshBrush allows you to paint meshes onto your GameObject's surface.

To start painting you need to select your GameObject on top of which you want to paint meshes in the scene, then go to the GameObject menu above in the toolbar and click on "Paint meshes on selected GameObject" inside the MeshBrush menu.

Note that the script doesn't work (unless in global painting mode) if you don't have a collider on your GameObject, so make sure you have one attached.

If you don't have a collider attached, MeshBrush will ask you if you want to add a Mesh Collider now. You can either say yes please and have a mesh collider attached, or if you want some other type of collider on your mesh you can say no and add one by yourself.

Now that you have added the MeshBrush script to your object, you will be able to hover over your GameObject with your mouse and already see the actual circle brush appear on your collider's surface.

Note: never have more than one collider on a GameObject on which you want to paint meshes: MeshBrush will only consider the first one you defined.

3. Usage and inspector

You can increase or decrease the brushes' radius by pressing or holding the I and O key on your keyboard. In case you can't see your brush on the object because of a near bright white light, you can change the default white color of the brush right in the inspector to help yourself see and organize your circle brush(es) better. If you don't like the default keyboard shortcuts, you can always customize them through the 'Customize Keyboard Shortcuts' foldout in the inspector.

To start painting meshes you have to assign at least one mesh prefab to the "Set of meshes to paint" array in the inspector first (more on that in chapter 3.2 “The set of meshes to paint”).

Now you can paint your mesh by hovering your mouse over your GameObject in the active scene view and pressing the P (like "paint") button.

This will place the single mesh in the center of the brush area and adapt its position and rotation to the underlying surface (the local Y-axis of the painted mesh will be the one pointing away from the surface of your GameObject).

You can also press and hold down your paint button: this will continuously paint your meshes until you let go of the button. The speed at which you paint meshes whilst holding down the P key is defined by the "Delay" value in the inspector, which sets the amount of seconds between brush strokes.

There is the possibility to delete the meshes you have painted, and the default key for that is L, but you can assign any key to that function in the inspector. The delete function will erase all painted meshes inside your circle brush's area. This is undoable, so you don't have to worry about an accidental mesh deletion.

There is a slope influence percentage slider that allows you to define how much sloped surfaces will affect the rotation of your painted meshes. A value of 100% for instance would align your painted meshes to be perfectly perpendicular to their underlying surface, whereas a value of 0% would keep your meshes either perfectly upright or tangent to their underlying surface (depending on whether you have checked or unchecked the "Y-Axis tangent to surface" toggle).

When painting upside down (like for instance grass growing out of a sloped ceiling or so) you may want to flip your painted meshes on their local Y-Axis if you are using rather low slope influence values. You can do this very easily by checking/unchecking the Invert Y-Axis toggle in the inspector view. This way YOU have all the control over your painted meshes, and you can make sure to have your painted meshes always point in the right direction.

New in version 1.5:

You now have a global paint mode, with which you can make use of MeshBrush without a collider. You can just paint your meshes directly into the scene. So if you choose this way over the traditional MeshBrush component way, just open up the GameObject menu and under MeshBrush click on Global painting.

This will automatically place a MeshBrush entity set up for global painting in the scene. You'll notice that the inspector is different compared to the traditional MeshBrush: there is a *layer based painting* section on the top.

When painting meshes globally into the scene, you can still have control over where you want your meshes to be placed and where absolutely not to. You achieve this with layers: just deselect all the layers in your global mesh painting inspector where you don't want your meshes to end up.

Note: The built-in *Ignore Raycast* layer is an exception in this case, because it really does ignore your raycasts: even if you have that layer active in global painting mode, meshes won't be placed there. It's because MeshBrush uses raycasting to paint meshes.

3.1. Templates

MeshBrush V1.5 comes with a brand new template system. It allows you to save your favourite MeshBrush settings into useful template files that you can later reuse in your scenes and projects. So if you find one of your MeshBrush configurations particularly worth saving, just do so by clicking on the save disk icon in the templates menu inside the inspector. You can then choose where to save your template file: it can be stored anywhere inside your project folder.

To load a template, press the load button right next to the save button and select the template file you want to load up. The current settings you have will be overwritten by the loaded template.

You can even save your favourite templates inside the favourites list in the inspector. Note that deleting, renaming or moving a template that has been added to the favourites list can break the connection between the two, and you'll have to reassign the path in the inspector. You can click on the three dots next to the entry in the favourites list to do so.

3.2. The set of meshes to paint

You can paint multiple meshes of the same type by simply adding one single mesh to the Set of meshes to paint and increase the number of meshes value in the inspector.

By increasing the *Nr. of meshes to paint* value, the scattering percentage slider will appear. Scattering defines (in percentage) how far the painted meshes are randomly scattered away from the center of your circle brush; so a value of zero for example would horribly spam all the meshes focused to the center of the circle brush area.

A value of 100% would use the entire brush area and eventually scatter your meshes even to the outer edge of your circle brush.

You can also paint many meshes of various types by adding more prefab meshes to your Set of meshes to paint. MeshBrush will then cycle through your set randomly and spawn the selected number of meshes (defined by the Nr. of meshes value in the inspector) on your GameObject's surface based on the Offset, Scattering, Randomize and Additive scale parameters. Since version 1.3 you can define a range within which a random number of meshes to paint will be chosen. To do so, just turn the *Use random number* on in the inspector.

New in V1.7: you can define a custom mesh density value (via the constant mesh density toggle in the inspector) to be used instead of a traditional fixed number of meshes to paint. This will keep the density of your painted meshes homogeneous, even when painting with different brush sizes. The mesh density unit is meshes/m^2.

3.3. Groups

Painted meshes can be further organized via group names (the defined group name is also the name of the parent holder GameObject).

There is always one group per MeshBrush component, so if you want to have multiple groups, just add more than one MeshBrush instance to your object. The name of the group can be modified at the top of the component's inspector (the changes will take place immediately).

Multiple groups per object means multiple brushes: you can help yourself and assign the various brushes different colors, this way you can differentiate between them when painting. You can disable a group by disabling its MeshBrush instance via the corresponding *Enabled* toggle in the inspector.

For instance: if you have a terrain mesh with two MeshBrush instances on it (one that paints meshes from a set of stones, the other one paints bushes), you can disable the stones group and only paint bushes. It's that easy!

Note: It is not recommended to have multiple groups on the same object sharing an identical name, as that can cause unexpected behaviours.

4. Features & functionalities

4.1. Precision placement mode

The precision placement mode allows for very accurate and reliable placement of one single mesh into the scene. Once active, the precision placement mode will override the default circle brush and show a preview mesh at the current mouse cursor's location. This entire procedure is divided into 4 simple steps:

1) **Location**
   Here you can choose which mesh to place (you can cycle through the meshes in your set with the N and B keys; N selects the next mesh in your set, whereas B cycles backwards) and confirm the desired location of your mesh directly in the scene view by pressing the paint button.
2) **Scale**
   This step lets you adjust the scale of your mesh by dragging your cursor away from (and to) the object's center.

3) **Rotation**
   Drag your mouse cursor horizontally to adjust the object's rotation along the Y-Axis.
4) **Offset**
   Drag your mouse cursor vertically to adjust the object's offset along the Y-Axis.

Confirming the vertical offset automatically places the mesh into the scene "as is", terminates the placement mode and returns back to the standard mesh painting mode.

4.2. Slope filter

With MeshBrush 1.2 a few new features came out like the slope filter for example.

You can use the slope filter to avoid having meshes painted onto slopes and hills and other steep areas of your GameObject's topology.

Adjust the slope filter maximum angle value to define the limit threshold angle above (or below) which no meshes may be painted.

You can also paint EXCLUSIVELY on slopes and hills; just invert the filter by toggling the *Inverse slope filter* toggle in the inspector.

The default reference vector used to calculate the slope angle on your painted meshes for the slope filter is the world's Y up vector.

In most cases, this is what you'll need (on terrains and other "flat" surfaces... and by flat I mean aligned with the world's XZ plane).

But there are a few scenarios in which you would want to use a different directional vector as a reference for the slope filter, like when you are painting onto round shaped meshes, e.g. planets and such, or simply painting diagonally/upside down.

4.3. Custom reference vector sampling

In these scenarios you can manually sample a reference vector directly from your GameObject's topology; to do so, just enable manual reference vector sampling in the inspector and activate the vector sampling mode by clicking on the button on the right.

This will switch your brush to the reference vector sampling mode: in this brush mode you can't paint meshes, you can only sample a vector from your mesh by pressing the regular paint button, or cancel the sampling process by hitting the Escape button. Deselecting and reselecting the GameObject will also cancel the sampling mode.

Once you have sampled a new reference slope vector, it will appear in your scene view where you sampled it as a visual reference arrow (unless you don't want that: in that case just turn it off in the inspector via the *Show sampled vector* option toggle).

The slope filter will now use that vector instead of the standard world Y up axis.

4.4. Randomizers

The randomizing parameters randomly alter the generation of the meshes you paint.

In the *Randomize* section of the inspector view you can randomly scale, rotate or offset the meshes you paint.

You can choose to randomly scale along all three X, Y and Z axes uniformly, or randomly scale along each individual axis.

Then there is the Scale within range toggle, which allows you to define custom ranges that MeshBrush will consider while randomizing your painted meshes.

Say we are painting for example a simple plain rock mesh onto a terrain (or also a simple crate onto a floor geometry mesh): first of all make sure the pivot point of your meshes are correctly set up. (For instance if you paint vegetation grass, the pivot should be placed at the bottom area of the mesh since that's where the meshes will sit on afterwards).

Now... if we choose to scale uniformly and not within a custom range, the Random Scale slider will appear. Let's say we put this slider to the value of 0.5... This will spawn the mesh smaller than the original prefab's size, and at the same time randomly scale it within a number near to that value of 0.5. This way, you can define not only how much randomness you want to apply, but also how much bigger or smaller you want your mesh to more or less be than the original prefab's size (you get two birds with one stone).

But what if this isn't enough control for us? What if we want to paint the mesh a bit taller than the original prefab mesh, but still more or less half the size?

No problem, just toggle on "Scale within range" and untick "Scale uniformly". Now you see four fields appear (an X, Y, Z and a W value). The X and Y values define the minimum and maximum width of the mesh (on the local X/Z axis of the mesh), while the Z and W values define the minimum/maximum height (Y-axis) of the painted mesh. So in our case, we could set something like X=0.4; Y=0.6; Z=0.5; W=0.9 to paint the mesh slightly taller than the original, but still randomly scaled between the minimum/maximum width and height.

You can freely combine the two Uniform scale and Range toggles (meaning you can ALWAYS at any time choose to scale uniformly and/or within custom ranges).

You can always delete all the meshes you've painted so far with the Delete button below in the inspector. Just fiddle around with the settings until you get a pleasant paint brush for your GameObject :)

Last but not least, you can randomly rotate the painted meshes on their local Y-axis with the Random Y rotation amount slider (value is in percentage).

E.g. a value of 0 would always paint your meshes with the same default Y rotation, and a value of 100% will always spawn your meshes with a different rotation on the Y axis.

4.5. Overlap Filter

MeshBrush V1.6 includes a new filter that avoids your painted meshes to overlap with each other. It's been added under the name of Overlap Filter, and can be found in MeshBrush's inspector right below the Randomize foldout.

To activate the overlap filter, tick the "Use overlap filter" toggle in the inspector and make sure the minimum absolute distance [m] value is greater than zero. This number indicates what absolute minimum distance (in meters) shall be kept and maintained between your painted meshes. The distance is measured between the painted meshes' transforms. Meshes closer to each other than the allowed min. absolute distance value won't be painted.

You can choose to pick a random value for the minimum absolute distance field: just tick "Random value" in the inspector and a random number between X and Y will be chosen for each mesh to paint in one paint stroke. E.g. if you paint 10 meshes at once per paint stroke, a new random min. distance value will be chosen 10 times (one for each mesh).

4.6. Additive scale

You can also choose to add an additive scale to your mesh below in the inspector, this will add the value you define to the painted meshes AFTER they have been randomized.

You can also additively shrink your models by simply typing in a negative number (which obviously can't and shouldn't go below -1... as we don't want to invert scale our painted meshes... That'd be weird :S

As in the random scale parameters, you can choose here too to scale uniformly on all three axes or along each axis individually.

In the next chapter, we'll cover optimization.

5. Optimization

When you are done painting meshes, you can optimize your scene by flagging what you've painted so far as static, or even by combining everything into one single mesh (per material).

You can also combine only the meshes inside the circle brush area; to do so, just press the combine button (default is K) and all painted meshes inside the brush area will get combined.

Combining the meshes will discard all of their components except their mesh renderer and filter, and automatically flag the combined mesh as static afterwards.

The combined mesh is stored by default inside the scene's root, but you can move the mesh data to the project folder and store it there for later re-usage. When you combine the meshes, the new combined object gets automatically selected (you can turn this off via the inspector toggle at the bottom) and gives you the option to save the mesh to disk.

If you want to re-use your combined mesh in other scenes (e.g. via a prefab) you have to save it to disk first, and then you can make a prefab out of it.

WARNING:

Combining meshes cannot be undone, so be careful with this button.

Only use it when you are really, really 100% sure that you're done painting meshes on your GameObject (or in a specific area of your GameObject); consider it a final "optimizing touch" to your scene.

There are detailed tooltips available for almost all settings in the inspector: just hover your mouse over the text labels for a few seconds and some useful infos will appear.

I also added an extensive "Help" foldout menu to the inspector of this tool for further information about MeshBrush's functionalities and tips for usage... or also just in case you forget your keyboard shortcuts.

Cheers ;)

Raphael Beck, 2015