[vPaint]

Advanced Vertex Painter for Unity3D

Features

- Fast and robust painting tools
- Photoshop style layers and blend modes
- Adjust and preview layers individually
- Paint your whole scene without breaking instances
- Blend colors automatically between objects
- Project lighting and textures onto vertex colors
- Group objects together to animate during runtime
- Shader library to allow you to utilize vertex colors

1. Introduction

Welcome to VPaint, the premiere vertex color tool for Unity3D. Vertex colors are a great way to add an extra level of visual polish to your game without the memory cost of additional textures. Games that target mobile devices have strict memory constraints, and vertex colors have very little memory overhead.

Vertex colors also add a lot of variation without the cost of an additional draw call. The vertex colors are uploaded to the graphics processor “for free” and can be used to blend multiple textures across one mesh, store lighting information, and control individual shader features.

VPaint also comes with features like texture projection and object blending, which are integral to creating a fast workflow. Lightmaps can be transferred from textures into the vertex colors in seconds, and a user can paint a single ground plane and project the colors onto objects sitting above it – saving hours from manual color blending.

2. Setting Up Your Scene

Setting up the vertex painter is a simple process. Follow these steps to get your scene fully prepared to accept and display vertex colors:

1. **Apply supported material to your objects**
    - To see vertex colors on an object, it must have a shader which supports vertex colors.
    - VPaint comes with a variety of such shaders, found under the VPaint shader category.
    - The most basic shader that supports vertex colors is VPaint/Unlit/VertexColorsRGB. A material can be found for this in VPaint/Materials/VertexColorsRGB.
    - You can also preview your vertex colors on the objects in a group by selecting the “RGB” button at the top of the VPaint window.
2. **Create a VPaint Group from your desired objects**
    - Select objects that you would like to paint. Objects must have a Mesh Filter and a Mesh Renderer to be paintable.
    - Select GameObject → VPaint → Create VPaint Group. This will create a VPaint Group based on your selection.
    - VPaint Groups are where vertex colors are stored. You must have the VPaint Group selected to paint on the objects.
    - VPaint will open automatically after creating a VPaint Group. To open it again, press “Open VPaint” in the group's inspector.

3. Add any additional objects to your VPaint Group

- In the Objects tab towards the bottom of the VPaint window, click the arrow in the top right corner and select “Add Object(s)”.

- Use this hierarchy window to select the objects you would like to add.

- Click on the object to select it,
- Shift-click it to select multiple and deselect
- Selecting a parent object will add all paintable children

- When you have all your objects selected, click “Add”.

4. You're ready to paint!

3. Painting Controls

This section will describe the options in the Color Palette panel and the Tool Settings panel, and their effect on the VPaint workflow. Here are a few important notes about the paint controls:

- All options only apply to the currently masked objects. For more information on this, see the Objects section.

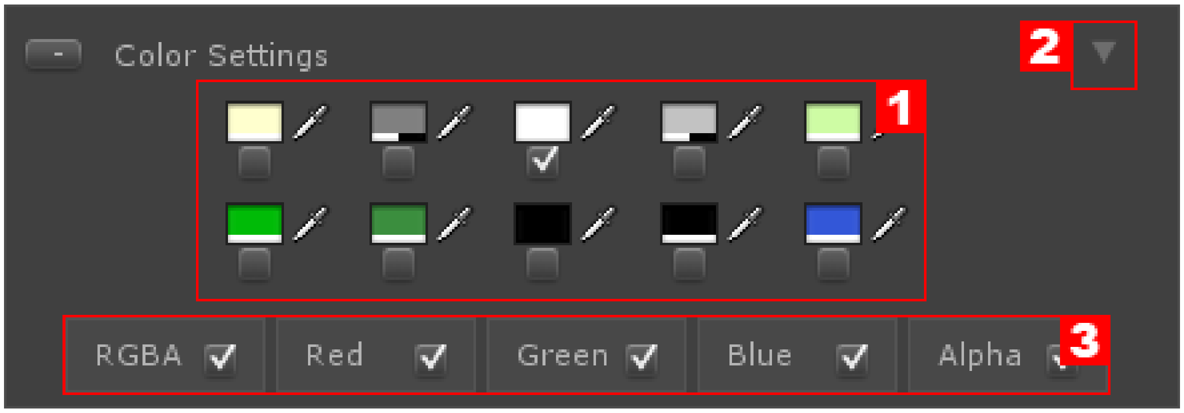


Figure 3.1

1. Color Palette
- The color palette stores 10 persistent colors.
- The color with the checked box below it is the currently selected color.

2. Functions
Flood Fill
- Floods all masked objects with the currently selected color.
- Flooded objects will become fully opaque.

Invert
- Inverts the colors of all masked objects.

Erase All
- Erases all color data stored on the masked objects.

3. Color Mask

- Color mask allows you to disable certain channels from your current brush.
- For example, if you want to only smooth the red channel, you can deselect Green, Blue, and Alpha.

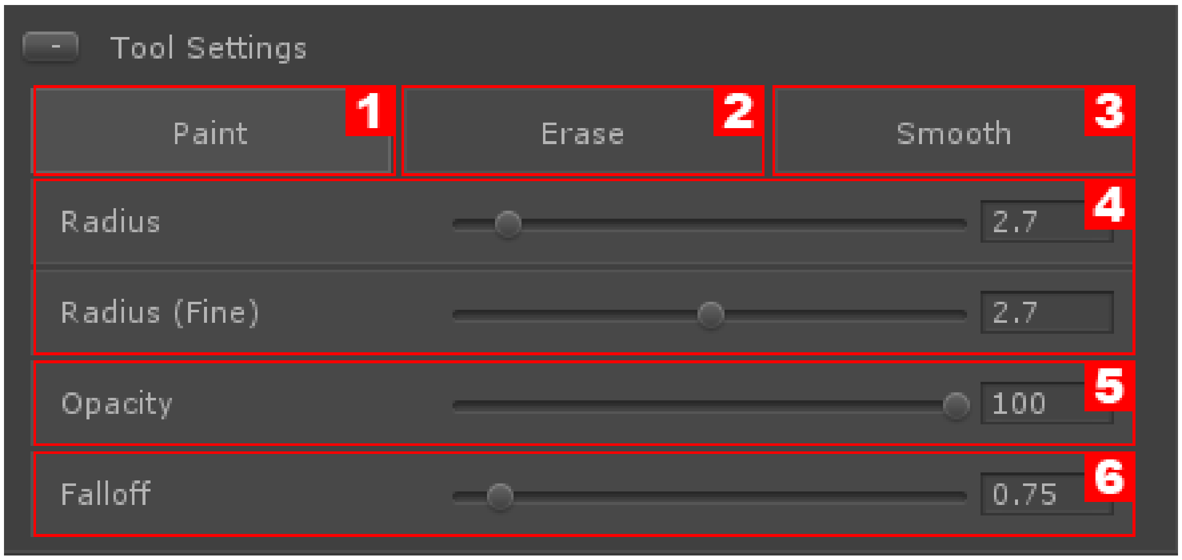


Figure 3.2

1. Paint Brush

- Adds the color selected in the Color Palette to objects.

2. Eraser Brush

- Reduces transparency on objects, erasing them from the layer.

3. Smooth Brush

- Smooths the colors on objects on the current layer.
- Note: This brush can be slow on dense meshes.

4. Radius

- The Radius option changes the size of the selected brush.
- The Radius (Fine) option modifies the Radius option in smaller increments. Some scenes are different scales than others, and sometimes you need to make very minor tweaks to the radius to achieve the desired effect.

5. Opacity

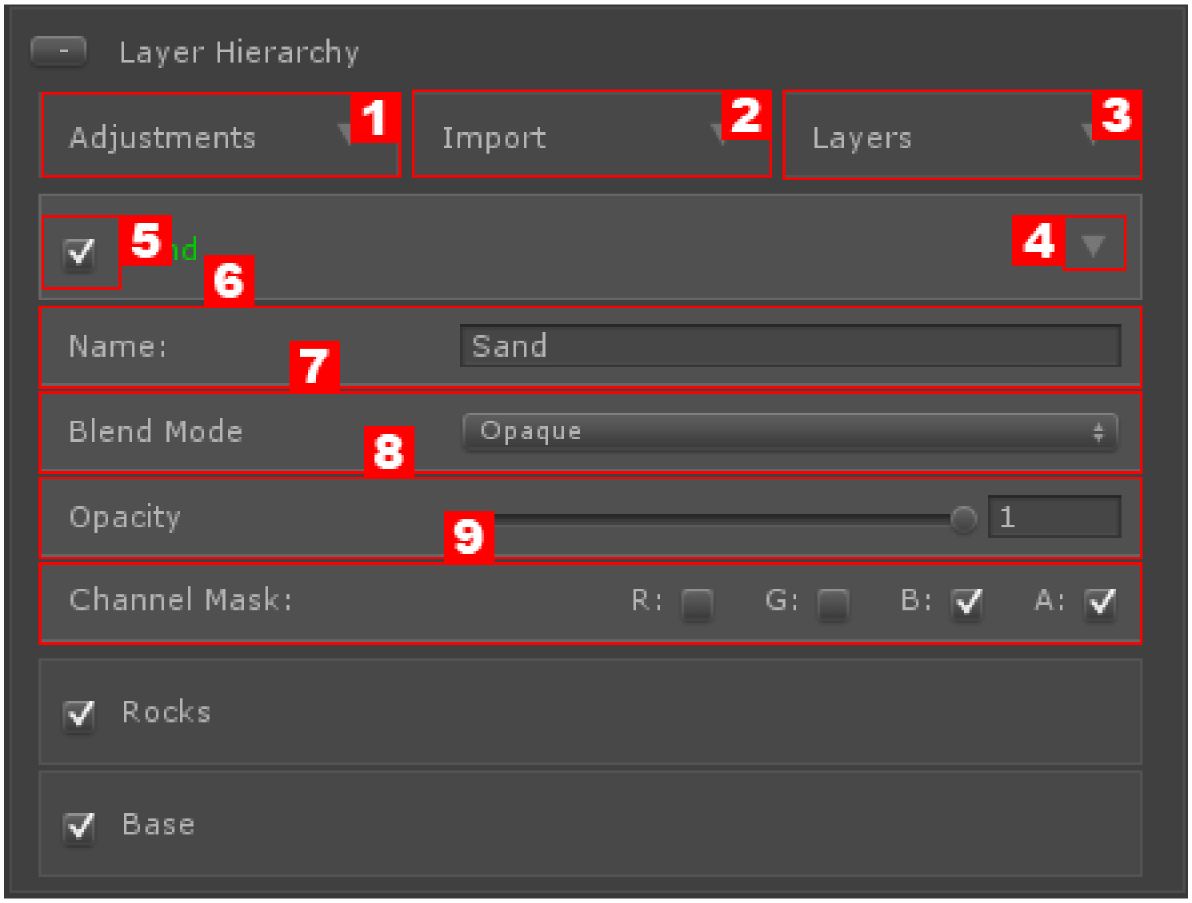
- The opacity to paint the vertex colors with. Lower values will apply less color over time while painting.

6. Falloff

- Exponential falloff of the brush. This is similar to hardness in Photoshop.
- 0 indicates that the brush has a completely hard edge.

4. Layer Hierarchy

Layers in VPaint are used to define aspects of the vertex colors which need to be modified separately. Photoshop users will be very familiar with this concept, and layers in VPaint have many similar features. Below is an outline of the Layers panel and what each button does.



Name: Sand

Blend Mode: Opaque

Opacity: 1

Channel Mask: R: ☐ G: ☐ B: ✓ A: ✓

Figure 4.1

1. Adjustments

Adjustments is a sub menu that gives access to functions which modify the currently selected layer. For more information on adjustments, see section 5.

2. Layers

The Layers menu provides options for changing the layer hierarchy.

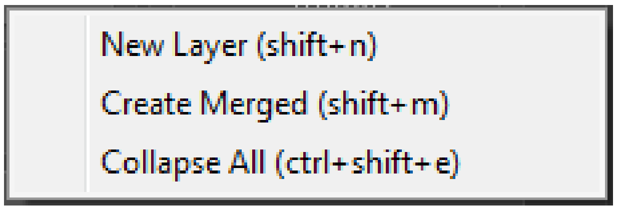


Figure 4.2

New Layer: Adds a new layer to the top of the layer hierarchy.

Create Merged: Duplicates all layers and collapses them into a single layer.

Collapse All: Collapses all layers into a single layer.

3. Import

The import menu allows you to import vertex colors from other sources. For more information on importing, see Section 6.

4. Layer Options

Clicking this arrow will provide options for modifying the selected layer. This menu is also accessible by right clicking on the layer.

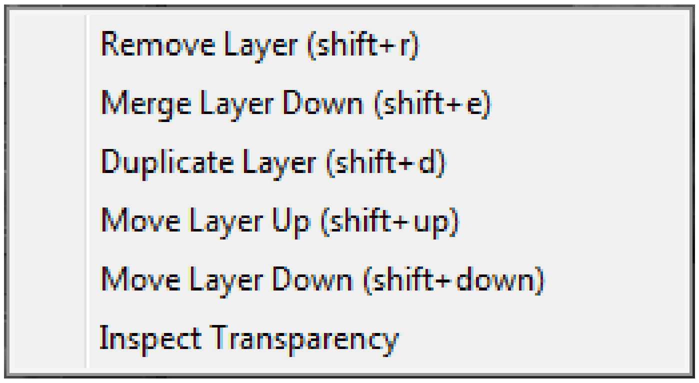

Remove Layer (shift+r)
Merge Layer Down (shift+e)
Duplicate Layer (shift+d)
Move Layer Up (shift+up)
Move Layer Down (shift+down)
Inspect Transparency

Figure 4.3

Remove Layer: Removes the layer from the layer hierarchy.

Merge Layer Down: Merges the layer with the layer immediately below it.

Duplicate Layer: Creates a copy of the layer.

Move Layer Up: Moves the layer up in the hierarchy.

Move Layer Down: Moves the layer down in the hierarchy.

Inspect Transparency: Renders the transparency of this layer to every channel of every object painted in this layer. This has no affect on the layer and is used only as a way to debug your layers.

5. Enabled Status

This check box indicates whether this layer is enabled or not. Layers that are disabled are never drawn to the vertex colors. Disabled layers cannot be painted on.

6. Name

Give the layer a name to identify it more easily.

7. Blend Mode

Blend modes allow one layer to modify the layers beneath it.

Opaque: Opaque layers render their colors on top of layers beneath them as is.

Multiply: Multiply layers multiply their colors against the layers beneath them. For example, a color of White will not affect any layers beneath them, but a color of 50% gray will halve the brightness.

Additive: Additive layers add their colors to the layers beneath them. For example, a color of Black will not affect any layers beneath them, but a color of 50% gray will increase the brightness by 2.

8. Opacity

The opacity setting allows one layer to be partially transparent to the layers beneath them.

9. Channel Mask

The channel mask allows the layer to isolate individual channels to be used in the layer rendering process. In Figure 4.1, only the alpha channel will be modified by this layer.

5. Adjustments

Adjustments allow layers to be modified as a whole, changing their overall features. Photoshop users will be familiar with many of the adjustments, such as Hue adjustments, Saturation adjustments, and Contrast adjustments. Adjustments affect only the selected layer.

To access adjustments, right click on the selected layer, or click on the “Adjustments Button” as shown in Figure 4.1.

All adjustments respect the masking options found in the Objects section and in the painting section.

1. Hue Shift

The HueShift adjustment modifies the overall Hue of the layer. It is measured in degrees.

2. Saturation

The Saturation adjustment allows you to desaturate or supersaturate the layer.

3. Brightness

The brightness adjustment allows you to adjust the brightness of the layer without destroying the hue.

4. Opacity Adjustment

The opacity adjustment allows you to modify the overall transparency of the layer. This is different from the opacity slider because you are modifying the actual transparency channel of each vertex, rather than the overall transparency.

5. Contrast

The contrast adjustment allows you to modify the contrast of this layer. The threshhold value allows you to change how wide of a range of contrast you affect.

6. Tint Color

The tint color adjustment allows you to apply a overall color to the entire layer. This is similar to flood fill, except that you have an option for opacity. You can additionally use the Mask By Value feature to apply the modification only to the upper or lower ranges of the layer.

7. Blending

The blending adjustment allows you to blend a set of objects to another set of objects. This is useful when you want to ensure that mesh seamlessly blends with another object.

There are two types of blending that can be used. The first is Directional, which is the most commonly used type of blending. This works by taking a directional sample from the target mesh at each vertex. The second type is Radial. This takes a sample from all vertices within a radius of each vertex. This type is less accurate and can produce artifacts, but is necessary in some situations.

Object blending can work within a bounding area to increase performance. Larger bounding areas will increase the amount of calculation that must occur for blending to be performed. It is a good idea to place your bounding box precisely where your blending should occur.

Intensity: The intensity of the blend. An intensity of 0 will not blend at all, and an intensity of 1 will apply the blend completely.

Directional Blending

Direction: The direction to apply the blend in. This is a world vector. For example, to blend down you would use (0,-1,0) and to blend right you would blend (1,0,0).

Offset: The offset to apply to the vertex position when sampling. It is useful to offset in the opposite directly of the blend by a slight amount, so that any vertices touching the object they are blending with will still receive a sample.

Falloff: The falloff value of the blend. A value of 1 will be a linear falloff, a value of 0 will blend 100% where any samples are found.

Distance: The distance to blend. This is in world units and this value will likely be different for every operation.

Radial Blending

Radius: The radius from each vert to search for samples.

Blend Objects

The blending adjustment works on a Source->Target paradigm. This means that there are a list of objects which are intending to blend with a list of blend targets. For example, if you are trying to blend Cube 1, which has no vertex colors, to Cube 2, which is painted, you would add Cube 2 to the Targets list.

You can disable objects from being blended by checking the box next to their name.

6. Importing

Importing allows you to bring in vertex colors from other sources. A great example of this is importing lightmapping or a texture into the vertex colors.

Base Layer

This import option will import the base layer of this object or objects into the layer hierarchy. This will import the vertex colors which were applied in the 3d modeling application used to create the mesh. If the mesh didn't have vertex colors, it will import a layer flood filled with black.

VPaint Group

This option allows you to import vertex colors from another VPaint Group. In this menu you can select whether to import all layers or just a specific layer.

Lightmaps

VPaint features full Beast lightmap support. After baking your scene in Beast, you can import that bake into VPaint directly. Importing lightmaps share many of the same settings as importing textures, and these settings are described below.

Import Intensity: The intensity to multiple against the lightmap. Beast lightmaps are rendered and stored at a lower intensity than they should be displayed. The value necessary will vary based on your setup, and it is a good idea to play with this value to get the right import. The default and recommended setting is 8.

Textures

VPaint can import textures directly to the vertex colors.

Texture: Select the texture that you would like to import.

Tile: Select the tile value that you would like to import at. This is similar to how the tile value on a shader works. Warning: Tiling does not work on clamped textures.

Offset: The UV offset to apply to the texture when importing.

UV Channel: Which UV channel to use when importing this texture.

Textures & Lightmaps Shared Settings

Import Transparency: This is the transparency value to apply to the imported colors. The default is 1.

Pixel Radius: This is the sample radius of the texture. This can help reduce noise when sampling from a diverse texture. On lightmaps, this setting can produce bleeding because Beast does not automatically extrude their UV edges.

Blur Radius: This is the radius with which the imported colors should be blurred after import.

Blur Threshhold: This threshhold determines the contrast of the blur.

Channel Assignments: Sometimes when importing a texture, the channels are not arranged how you want them, or you want to discard whole channels entirely. In GPU programming this is called “Swizzling”. In the Channel Assignments section you can designate what to do with each channel. Clicking the “Reset” button will reset all channels to go to their original locations.

Target Layer: This setting determines whether to overwrite the selected layer or to import the texture as a new layer on top of the layer stack.

7. Object Targets

The Object Hierarchy lets you determine what objects are a part of the currently selected VPaint Group. Using the menus on the right, you can modify this hierarchy.

The check boxes on the left indicate whether this object is currently paintable. Disabling this will disable any operations from modifying this object. You can use this if you want to isolate your painting to one object, or if you want to selectively perform adjustments.

To add new objects to your VPaint Group, select the arrow in the upper right corner of the Object Targets tab. Then, select the objects you wish to add and click “Add”.

8. Isolated Region

The isolated region tab allows you to isolate a particular region to paint on. Working with a very large set of objects can cause the editor to slow down, and this will help manage the performance of the editor by isolating the calculations to one region.

This tool is also useful when trying mask your painting. When the isolated region is enabled, painting will only affect vertices contained within the bounding box.

There are two additional options provided for this tab:

1 Reset: Reverts the isolated region to the world origin

2 Focus: Places the isolated region centered on your scene camera

10. Code API

This section describes the commonly used methods to use in your code.

VPaintObject

List<VPaintObject> OverlapSphere

Returns all the VPaint Objects within a radius.
Position: The origin of the sphere
radius: The radius of the sphere

void ApplyPositionalModifier

Applys a positional modification to the VPaint Object's color set.
worldPosition: The origin of the modification
modifier: The modifier to apply

IEnumerator ApplyPositionalModifierAsync

This is the same as the ApplyPositionalModifier function, but is used as a coroutine.

void ApplyColorSpherical

Applies a color based on the distance from the object.
Color: The color to apply
WorldPosition: The origin of the sphere to apply from
Radius: The radius of the sphere to apply from
Falloff: The falloff of the color modification
Strength: The intensity of the color modification

void ApplyAlphaSpherical

This is the same as ApplyColorSpherical, but applies only to the alpha channel.

Here is a code example of how to apply a spherical modification to your scene:

```
Vector3 position = Vector3.zero;
float radius = 1;
float falloff = 0.5f;
float intensity = 1;

Color color = Color.blue;

List<VPaintObject> objects = VPaintObject.OverlapSphere(position, radius);
foreach(VPaintObject obj in objects){
        obj.ApplyColorSpherical(color, position, radius, falloff, strength);
}
```

VPaintUtility

void BlendRadial

- Layers: The layers to affect on this blend operation.
  - For example: new VPaintLayer[]{ vPaintObject.paintLayer }
- BlendObjects: The objects that you would like to blend
- BlendTargets: The objects that you are blending onto.
- Radius: The radius of the radial sample
- Intensity: The intensity of the blend modification
- Bounds: The bounding box to use (Optional)

void BlendDirectional

- Layers: The layers to affect on this blend operation.
  - For example: new VPaintLayer[]{ vPaintObject.paintLayer }
- BlendObjects: The objects that you would like to blend
- BlendTargets: The objects that you are blending onto.
- Direction: The direction of the directional blend
- Distance: The distance of the directional blend.
- Intensity: Intensity of the blend modification
- Falloff: Falloff of the blend modification (based on Distance)
- Offset: The offset to use for blending samples. (Optional)
  - Useful for avoiding problems when objects intersect.
- Bounds: The bounding box to use (Optional)

Async operations exist for both of the above blending operations.

IEnumerator LerpColors

- Used to interpolate between two different color sets.
- This method can be used as a coroutine.
- Obj: The object to interpolate colors on
- Start: The color set which the object should begin with
- End: The color set which the object should end on
- Time: The time the interpolation should take.

VPaintGroup

void Apply

Applies the VPaint group to its objects.

IEnumerator BlendTo

Blends this VPaintGroup towards another VPaintGroup.
Can be used as a Coroutine.
Target: The VPaintGroup to blend towards.
Time: The time the blend operation should take.

10. Frequency Asked Questions

What are vertex colors?

Vertex colors are a type of data that is stored on a mesh. Unlike textures, unique vertex colors can be applied to an object without requiring an additional draw call.

How does instancing work?

Usually to modify a mesh in Unity you must create a new instance, breaking the original reference to your model file. We thought this was a terrible workflow, since updating models in a game is important, and we have worked up a solution to fix this. In VPaint you can modify your meshes to your hearts content without worrying about instance breaking.

Does changing the vertex count or vertex order of my mesh break vertex colors?

Yes, currently if you are to update your mesh with a new version it will break vertex colors on any instances of that mesh. In a future update we will include a process for transferring vertex colors from the previous mesh onto the new mesh.

Where can I direct my questions?

We have a forum thread running at this address:

http://forum.unity3d.com/threads/192711-VPaint-Advanced-Vertex-Painting-Released

Additionally, you can contact the creator of this tool at beck@valent.us