PIXELATED POPE'S

RETRO PALETTE SWAPPER

MANUAL

PIXELATED POPE

Table of Contents

Requirements and Limitations

Requirements

Alright, first and foremost, let's talk about what this does to the minimum requirements for your project. The big one: You are using shaders now, and specifically Shader Model 3.0. This is not a new thing by any stretch; Nvidia began supporting it in 2004! So it is very likely that if your project is run on hardware that doesn't support shader model 3, at best your pal swap effect will not be visible, and at worst your game will crash on start. So, as this is technology that has been around for 15 years, what sort of hardware are we concerned about? Laptops. There are very, very few desktops out there in the wild that don't have some sort of GPU that can do shaders, but may laptops today have dual GPUs, an integrated card and a dedicated graphics card. If there is no dedicated graphics card, there is a good chance that your game will not work on it at all. And even if it does have a dedicated graphics card, there's a chance the game may be ran utilizing the onboard GPU rather than the dedicated GPU. This is often fixed using the GPU's setting manager, and the OS should be smart enough to recognize when to use the dedicated GPU, but unfortunately this isn't always the case.
Ultimately, this shouldn't be too big of a concern, especially if your project already has other shaders you plan on using. Only the absolute lowest end of hardware will no longer be supported.

Additionally, as the name of the shader implies, the art for your game should be pixel art based. Smooth, anti-aliased, high resolution art is not really compatible with this shader. So if you would have trouble counting the number of colors used in any one of your sprites, this may not be the shader for you.

As such, it is highly recommended that the "Interpolate Colors Between Pixels" global game setting be turned off for your project for all target platforms.

Limitations

While generally shaders are pretty damn fast, Retro Palette Swapper is unfortunately a bit more complex than some shaders. Essentially, the shader does this: For every single pixel drawn with the shader set:

1. Look at the ARGB value of the color being drawn.
2. Loop through each pixel on the palette sprite and compare it to the current pixel until a match is found.
3. Replace the original color with the color on the same row as the matched color, but at the indicated column.

Steps 1 and 3 aren't so bad, but that for loop can be pretty brutal. As such - and as the name of the system implies - you really shouldn't be using this on a TON of pixels. I'll discuss this a bit more in the section on applying this to your entire screen, but the more pixels you are drawing with the shader, the slower it's going to

be. This means if your game supports sub pixels, you need to be careful of how much you are using this. As even if your app surface is only twice the width and the height of your view, that means you are drawing 4 times the number of pixels! So, just be careful about that.
On that same note, having many different objects all managing their own palette swaps could cause some issues. The shader_set() and shader_reset() functions are actually pretty heavy. So if you are planning on using this for like... an RTS or something where there are over a hundred individual instances all setting the shader, drawing, and resetting... might need to do something fancy to reduce the number of calls to those two functions.

Summary

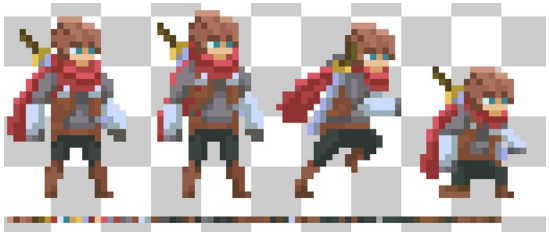
1. Raises min requirements to a GPU that supports Shader Model 3.0
2. Pixel Art Assets only
3. Interpolate Colors Option needs to be Off
4. Limit the number of pixels as often as possible
5. Limit the number of pal_swap_set/reset as much as possible.

Preparing Your Assets

As I've said many times already, we should only be dealing with pixel art assets here. Let's look at some examples:

This is the ideal situation. Every animation of this character across multiple sprites all use the exact same 15 or so colors. There is 0 deviation. You can have more colors, but making sure all colors are consistent across all sprites is vital. And keeping the number of colors down will always be a positive thing.

This is less ideal. While it's pixel art, the color palette is very muddy. Look at all those different browns and reds and greyish greens. It's pretty chaotic, and it will make designing alternate palettes that look good very difficult, as well as impact performance if many characters have similarly large, disorganized palettes.



This is right out. Really not a good idea to use art that is smoothed out like this. Could it work? Sure. Should you do it? No. Palette Generator will choke on this and die because it is well over 256 unique colors.

So the most important step of preparing your art for use with palette swapper is making sure that for any one group of sprites you want to swap (such as all the sprites for a single character or tileset) they all use the exact same set of colors, it will make creating your pal sprite

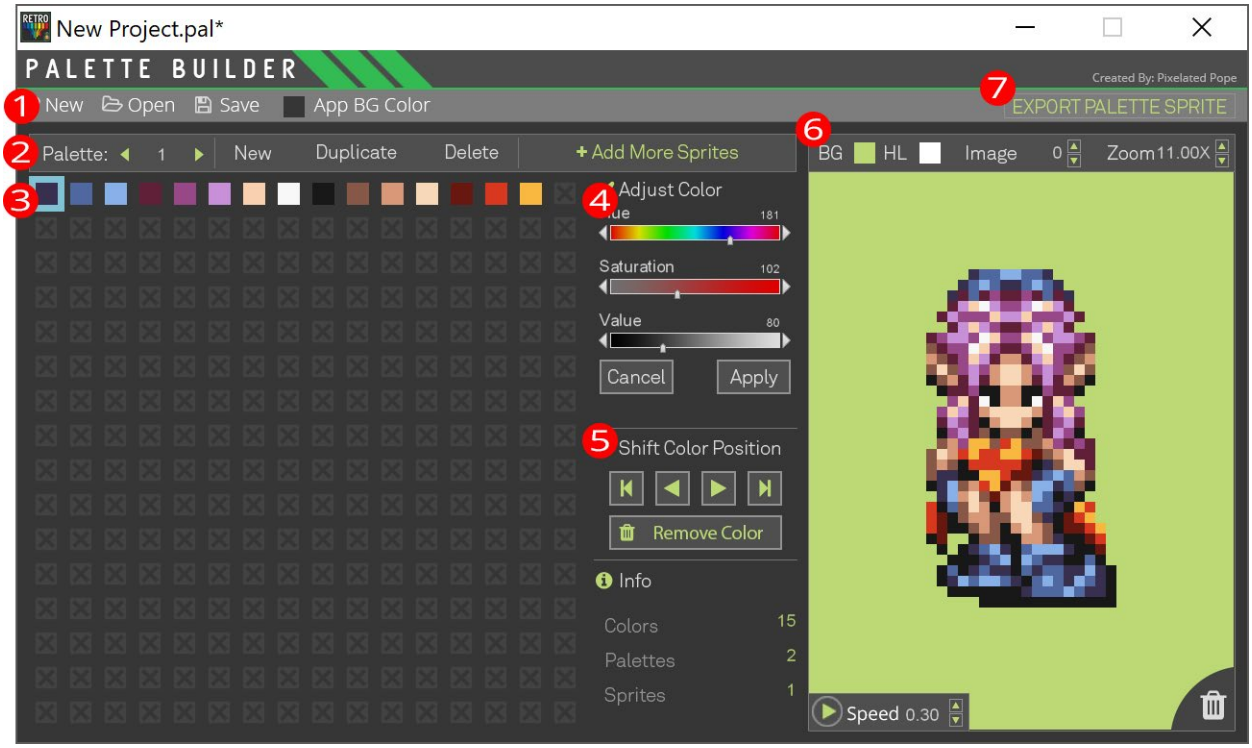
Summary

1. Make sure to make "similar" colors "exact" colors.
2. Same colors used across all related sprites.

3. No anti-aliased sprites. Retro only. It's right there in the name.

Using Palette Generator

Palette Builder is a tool that is useful for building your palette sprites and designing alternative palettes for your sprites. It's not the most intuitive software (it was made in GMS1.4), so let's go through the basic functions!



1. File Toolbar

New: Clear your current project and start fresh.

Open: Open a previously saved pal project.

Save: Save your current project. This will include all palettes and images that have been added.

App BG Color: You can use this to change the background color of the app. Just for fun.

2. Palette Toolbar

Palette Selector: This allows you to swap the currently displayed palette. You can modify any but the "base" palette.

New: Creates a new palette based on the base palette.

Duplicate: Creates a new palette based on the currently selected palette.

Delete: Deletes the currently selected palette. Can't delete the base palette.

Add More Sprites: Adds a sprite to the project. All the unique colors in the sprite will be analyzed and any colors not already in the base palette will be added to the project. If your sprite is a sprite strip, you can name it <name>_strip<number> to have the app split it up automatically for you. For example: spr_pope_strip17.png

3. Color Panel

The color panel shows all colors in the current palette. All colors in the base palette will be unique, but this is not necessarily true for other palettes. Colors can be selected by left clicking on them. Shift + Left Click will perform a range selection from the previously selected color. Ctrl + Left Click will add/remove colors from the current selection. Holding alt while mousing over a color will make that color blink in the Preview Panel. Holding spacebar will do the same thing, but for all currently selected colors.

4. Adjustment Panel

When any palette besides the base palette is selected, you can use the Adjustment Panel to modify all currently selected colors. Colors are modified using HSV. For all 3 bars, you can left click on the bar to move the current value. You can also left click the arrows on either side to make minor adjustments. You can also right click anywhere on the bar and type in your desired value.

Finally, you must apply any changes before changing your selection. This will commit all changes to all selected colors. This cannot be undone. The cancel button will discard any current changes.

5. Color Position Panel

You can use this panel to re-order the currently selected colors. Colors can be moved one spot left or right, or pushed all the way to the top left or bottom right. Reordering the colors will change the position of those same color positions in all palettes. Typically you want to organize your colors using your base palette. You can also use the delete button to remove a color from the palette. The only way to get this color back is to re-add the sprite that contained that color.

6. Preview Panel

The preview panel shows a sprite displayed with the currently se lected palette.

BG: Allows you to change the background color of the preview panel.

HL: Allows you to change the highlight color that shows up when holding space or alt in the color panel.

Image: The currently displayed image of all added sprites.

Zoom: Zooms the preview image in and out. Clicking on the arrows will change the zoom in the desired direction by .1x. Holding shift while clicking either button will change the zoom by 1x. Using the mouse wheel while your mouse is over the preview window will do the same thing, including the increased zoom speed when holding shift.

Preview Window: Shows the current sprite. You can left click and drag it around to reposition the image in the window.

Play/Pause: Will automatically cycle through every image at the given speed.

Trash: Will remove the current image from the project. This will not affect your current collection of colors in any way.

7. Export Button

When you are done creating all of your palettes, click this button to export your palette sprite. Add this sprite to your GameMaker project and make sure the origin is in the top left corner. This is the sprite you will use to make the actual pal sprite.

8. Hidden Functions!

Press the Z key to change the size of the app.

Press the X key to export all your added sprites using the currently selected palette. You must save your project first, and then all images will be exported to a folder next to your saved project file. This is very useful when used with the next hidden feature...

Press C to “crush” your palette. What does that mean? Well, remember when I said it was important that you don’t have lots of “similar” colors back in the section about setting your art assets up? This feature can help you make similar colors exact matches! When you press C, the app will ask you for a factor. This is a number between 2 and 255. But what is this actually doing? It will essentially round the RGB value of each color to the nearest multiple of the given factor. For example, if you have a color that is 123,56,22 and you crush it with a factor of 16, you’ll end up with an rgb of 128, 64, 16. Do this to many colors that are similar to 123,56,22 and they will all likely end up with the same RGB value. Every time you crush your colors, you’ll get a new palette added to your project, so experiment with many factors until you get the desired result. Once you get what you want, use the X key to export all your added sprites with the current palette. Create a new pal project with the exported sprites, and you’ll have a smaller color set to work with.

Your First Palette Swap

Alright. You've prepared your sprites, and created your custom palettes using palette generator... time to get everything up and running in your project. Fortunately, you've done the hard part already! So let's get this set up real fast.

Add the system to your project!

1. Shaders
    a. shd_pal_swapper - for most platforms
    b. shd_pal_html_sprite & shd_pal_html_surface if you are targeting HTML5
2. Scripts
    a. pal_swap_init_system
    b. pal_swap_set
    c. pal_swap_reset
    d. Optional Depth and Layer Swapping
        i. pal_swap_set_depth
        ii. pal_swap_enable_layer
        iii. pal_swap_set_layer
        iv. _pal_swap_layer_start
        v. _pal_swap_layer_end
    e. Optional Misc:
        i. pal_swap_draw_palette
        ii. pal_swap_get_color_count
        iii. Pal_swap_get_pal_count
3. Objects
    a. Optional for swapping based on depth
        i. Obj_depth_swapper_start
        ii. obj_depth_swapper_set

Initialize the system!

Simply call **pal_swap_init_system(shd_pal_swapper)** one time at the beginning of your game. This will initialize a number of global variables used by the system. If you've renamed the shader for some reason, you would pass that name instead of shd_pal_swapper.

Start the pal swap!

In your draw event, call **pal_swap_set(palette_sprite,pal_index,false);**. This will turn the pal swap shader on. The palette sprite is the sprite you got out of Palette Generator. The index is which column of that sprite you want to draw using (0 being the default palette). The third argument tells the system whether your palette sprite is a surface (true) or a sprite (false). Unless you are building a palette on the fly, it will usually be a sprite, so this argument should be set to “false”.

Draw!

With the palette set, all further drawing will be done with the desired palette. You can draw anything: a sprite, a surface, text, etc. As long as the exact colors used in the first column of your palette sprite are used to draw with, you’ll get the alternate colors.

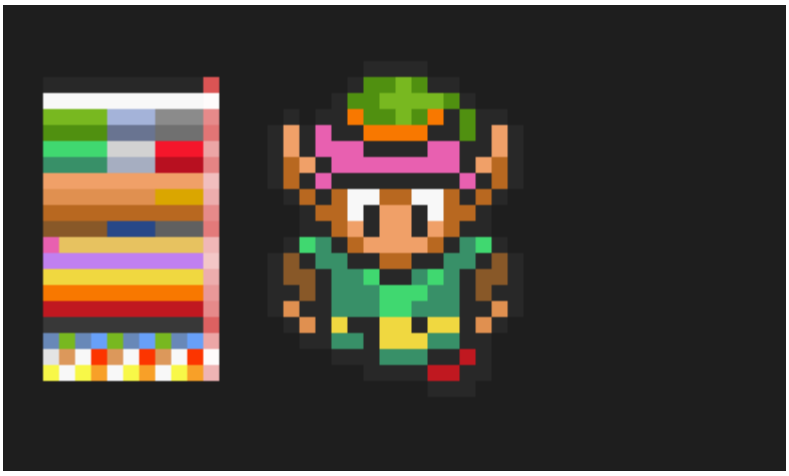
Reset!

When you are done drawing with your alternate palette, you need to turn it off again You do this by calling **pal_swap_reset();**.

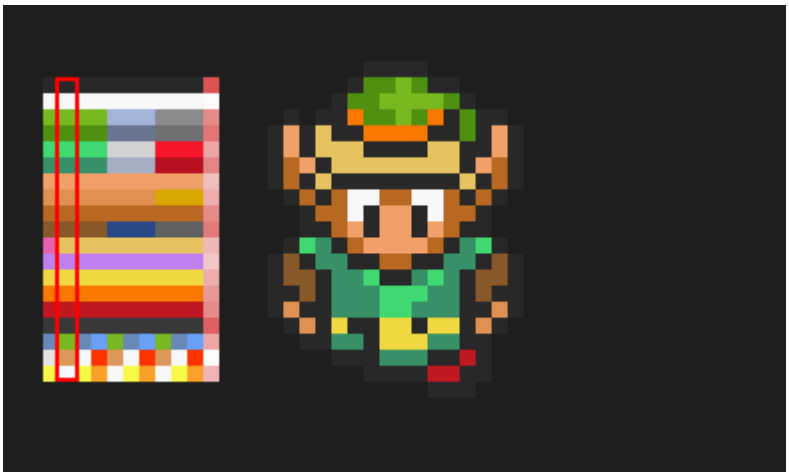
Dynamic Surface Palettes

Dynamic palettes are a powerful tool for creating dynamic effects using palette swapping. They can be a bit mind bending, but I'll do my best to explain them here.

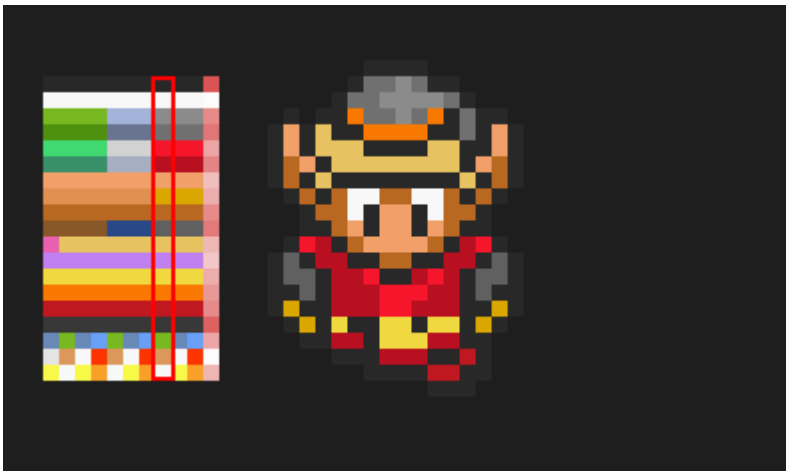
This is Link, from A Link to the Past.



Although in my old Zelda fan project, you'd ee him like this with blond hair. He's using Palette 1 here.



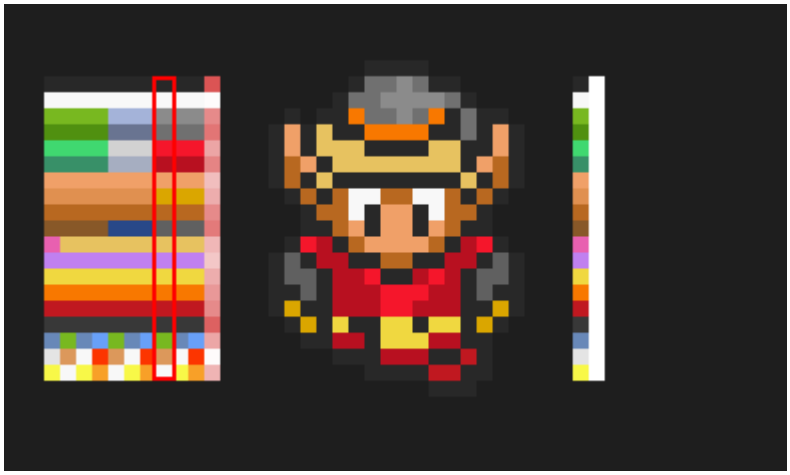
And of course, later in the game, he would look like this, using palette 7,8, or 9 depending on his sword.



But what if I wanted to change the color of his outline without modifying my palette sprite?

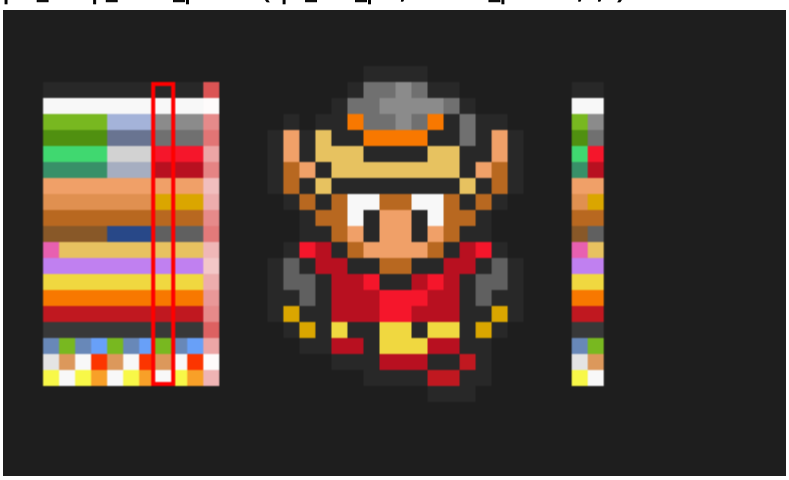
First, I would create a surface. The surface should be as tall as my palette, and 2 pixels wide.

On that surface I would call **pal_swap_draw_palette(spr_link_pal,0,0,0);** to draw link's base palette to the left column of the new surface. Just like a normal palette sprite should look.



Then use this same script again to draw his current palette.

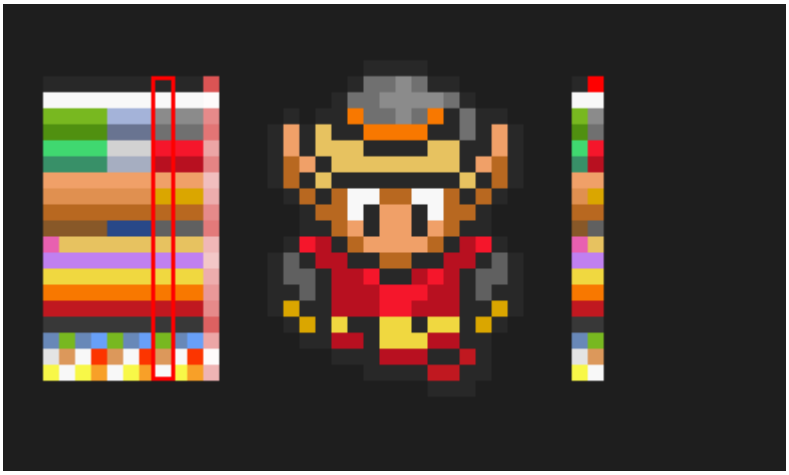
pal_swap_draw_palette(spr_link_pal,current_palette,1,0).



Now, to change the outline, simply draw over the outline's color on the surface. In this case, that is the color in position 0. Like so:

draw_set_color(c_red)
draw_point(1,0);

To get this:



But the palette isn't being used yet. Back where we are drawing link, we would need to pass the surface to the shader using pal_swap_set, and tell the shader that we are using a surface instead of our normal pal sprite. Since you likely not using the outline all the time, you would need a variable to toggle between outlined and not outlined. The full code may look something like this:

```
if(pal_override)
   pal_swap_set(override_surface,1,true);
else
   pal_swap_set(spr_link_pal,current_palette,false);
draw_self();
pal_swap_reset();
```

And this would be the result:

Anything is possible when drawing that second palette to the surface, including drawing it to the surface using ANOTHER shader such as grayscale, sepia, or infrared.

If you wanted to "fade" from the current palette to the custom palette, make the custom palette surface 3 pixels wide, duplicate the current palette into column 2 and 3, overwrite the outline color in column 3, and then gradually change the palette index value from 1 to 2. This is what it would look like halfway between 1 and 2:

Essentially you stop using your original palette sprite all together, and instead use columns from it to build a new palette sprite on the fly in a surface.

Full Screen Swaps

[Coming Soon]

HTML5 Support

[Coming Soon]

FAQ

[Coming Soon]