How to start making pixel art #1

An absolute beginner's guide

Pedro Medeiros Follow
Jul 21, 2018 · 9 min read

This is a little article on how to start making pixel art, intended for those who are really starting out or never even opened a pixel art software. For now I'll cover only the very basics, how to create a file, setup the canvas size, and work with a color limit.

This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

Also, this is the part 1 of a series of articles, read the whole series here in the Pixel Grimoire.

Before Starting

Before jumping into pixel art, remember: pixel art is just another art medium, like guache, oil painting, pencil, sculpture or its close cousin mosaic. To make good pixel art you need to be able to make good drawings. In general, this means studying anatomy, perspective, light and shadow, color theory and even art history, as these are all essential for making good pixel art.

Tools

You don't need anything fancy to make good pixel art, and you can do fine even with just a good mouse and free software. My setup includes a small Wacom pen tablet, a good mouse, a good keyboard and my favorite software is **Aseprite**, but you should use whatever your're most comfortable with.

Here's a list of software commonly used for pixel art:

- **Aseprite:** Great professional editor with many time-saving features (paid)

- **GraphicsGale:** A classic, used in many games. It's a little complex, but full of great features (free)
- **Piskel:** Free online pixel art editor (free)
- **Photoshop:** Powerful image editor not intended to make pixel art but you can set it up to use it (paid)

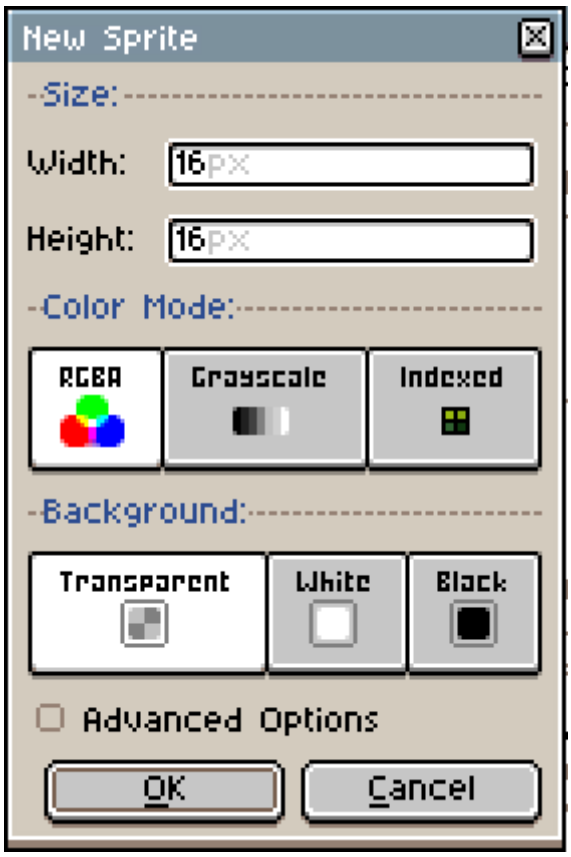
Aseprite

Aseprite is my favorite pixel art software right now. It's incredibly powerful, packed with features and yet simple to use. I chose Aseprite as the software for this tutorial but I'm pretty sure you can adapt it to any other software you use with minimum changes. You can also get the free trial for Aseprite, but keep in mind it won't save your files, which I guess it's OK if you are just practicing.

Making a New File

Just click the "**New File…**" link in the home screen or go to **File > New File** so we can start drawing.

Let's create a new file. 16 by 16 probably seems a little too small, but I think it's a good starting point. Bigger resolutions can distract you from what you should focus now: understanding the interactions of pixels with their neighbors.



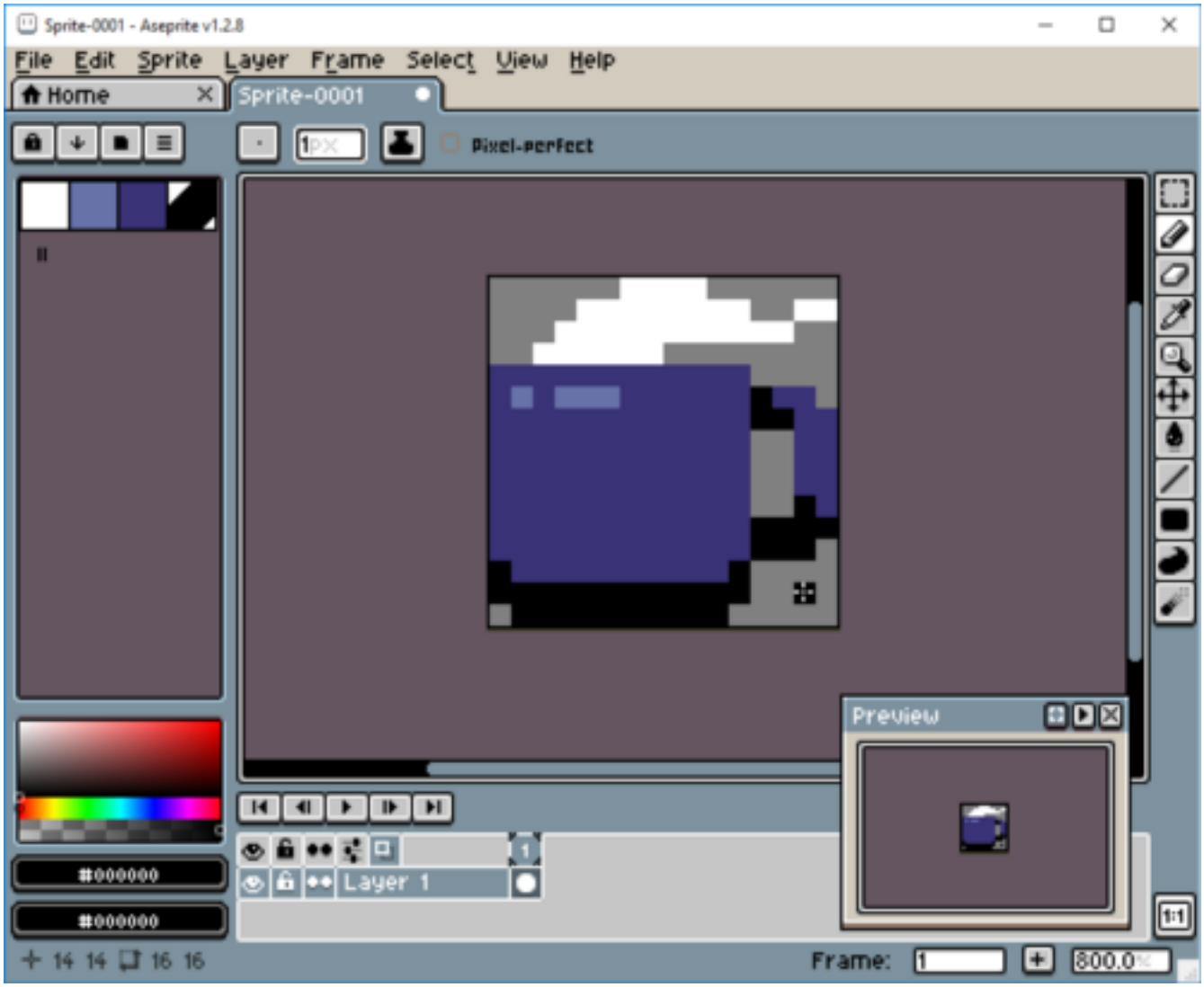
Aseprite 'New Sprite' dialog

You can leave the color mode in **RGBA**, that is the most simple and intuitive for now. Some pixel artists like to work with an **indexed** palette which allows some pretty cool color tricks, but comes with some drawbacks too.

Keep the background **transparent** or **white**, it won't change much for now. Just make sure that Advanced Options is unchecked (but feel free to experiment with them later) and you are good to go!

Let's Draw!

There are lots of toolbars and menus there, but don't worry, we just need a few buttons for now. The main tool is the **Pencil**, that should always be kept with 1 pixel of width, and it will be how we place our pixels on the canvas. Just click the button, or press **B**, and click on the screen to place down a pixel of the selected color.



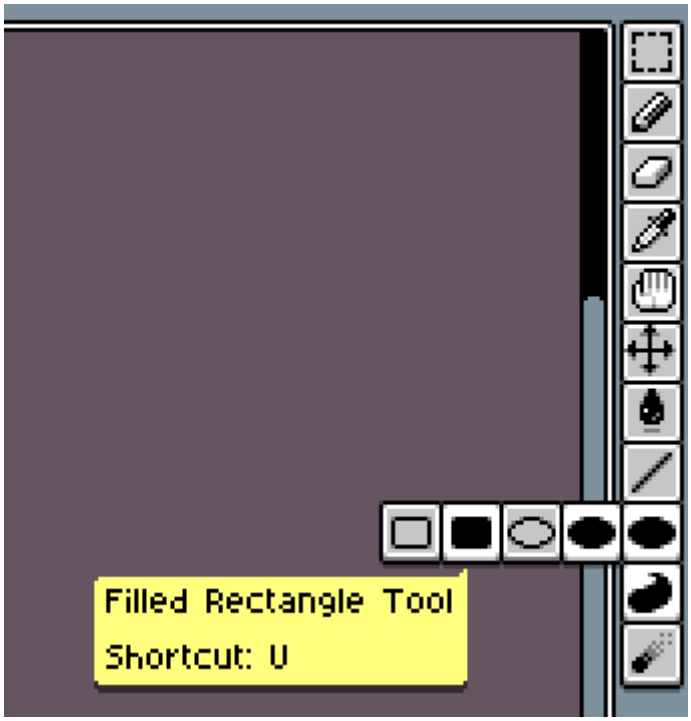
Aseprite workspace

On the left you can see your color palette, with some of the default colors. Let's change those to another, simpler set. Click on the third icon on top of the color palette (*Pressets*) and choose **ARQ4** (a really good palette made by Endesga), that's the one you will be using for your first sprite.

Now, only using the 4 colors on the top left, try drawing a **mug.**

Feel free to use mine as an inspiration, but also try making it unique. If you make a mistake, **alt+click** on an empty area or outside of your drawing and you will "pick" the transparent color and you can use it to erase pixels. Alternatively you can click on the **Eraser** or press **E** to select it.

You will probably notice that working in such a low resolution is very different from regular drawing. Everything needs to be calculated, and each pixel you place is a big choice you need to make. That's the thing you will need to get used to.



You can also experiment with the other buttons in the toolbar. It's worth noticing that some buttons will open more options when pressed. Just avoid the blur tool for now, as it adds more colors and we don't want that yet.

Next, let's make more sprites! Try drawing a **skull**, a **sword** and a **human face**. This time without my pixel art reference. If you feel that the sprites simply won't fit in the canvas, that's absolutely normal, try abstracting something to a single pixel and try again. It's very hard to work with such a low resolution and it feels like a puzzle

sometimes. Here's another article I wrote about working with low resolutions for Kano: [link]

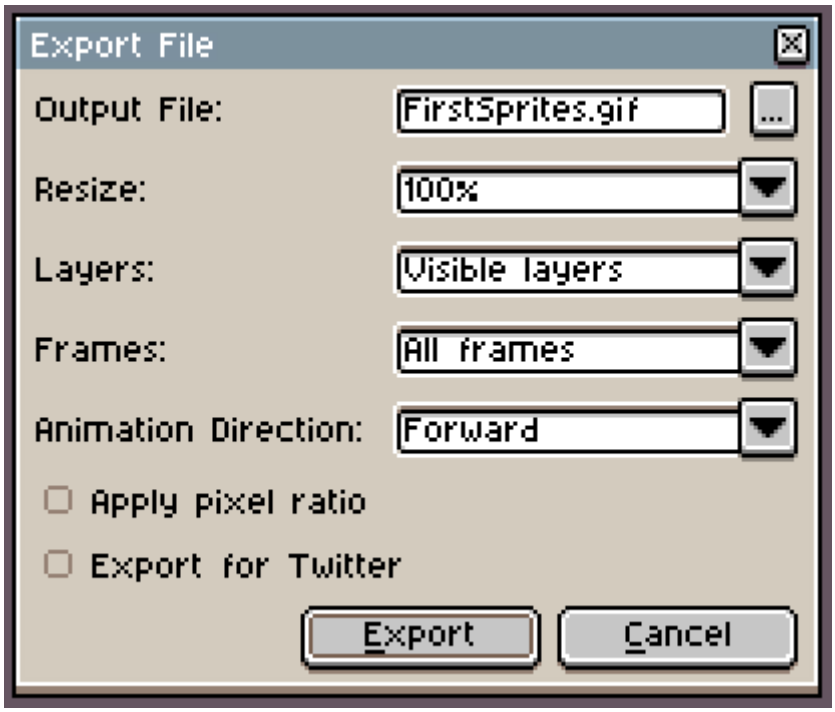
If you want, here's my versions of those sprites, just please make sure to finish yours before looking at them [skull, sword and human face].

This is always a good exercise. If you want to keep practicing, try making even more drawings with those constrains.

Saving Your File

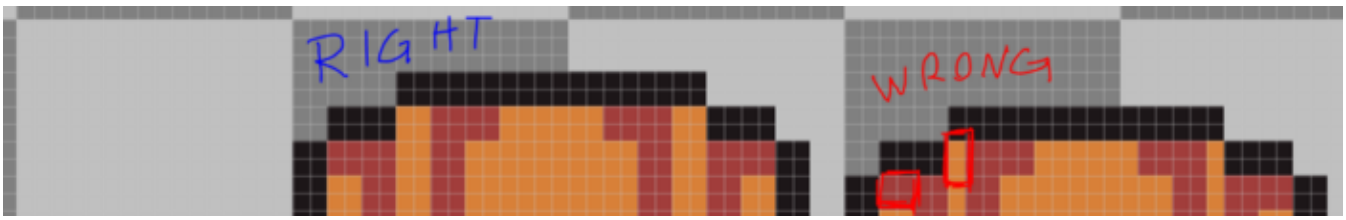
To save your file press **Control+S** (or go to **File>Save As…**), choose a file name and location and just hit save.

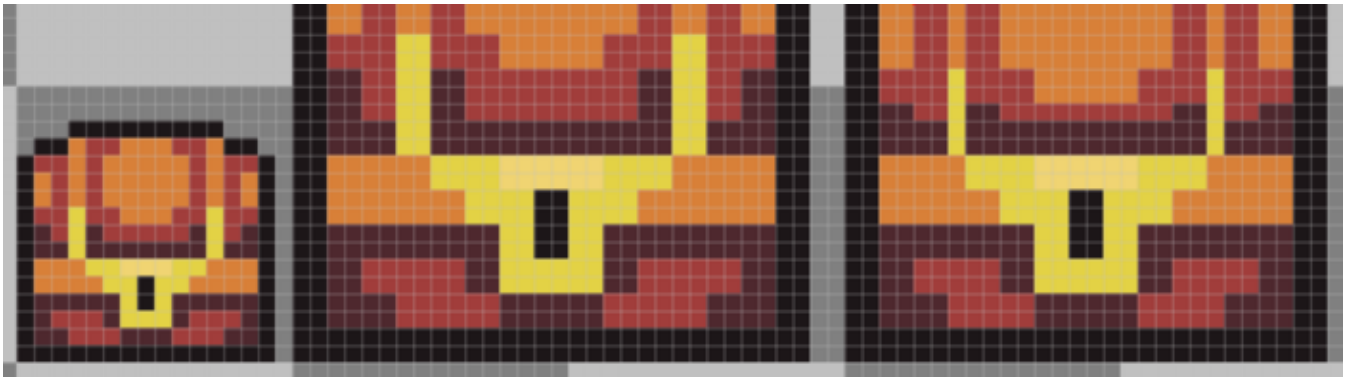
Don't forget that in the trial version of Aseprite saving is disabled!



Aseprite Export File dialog

You will see that Aseprite can save in a variety of formats, but I always recommend keeping a **.ase** version of every file you make. Just like in Photoshop you would keep a **.psd** file. When exporting for web or games, you can use **Control+Alt+Shif+S** or **File>Export**.





Why you should never resize a pixel art partially

Aseprite has this really good **Resize** feature in the export window. It only scales your sprite in round numbers, which is perfect. If you rescale your sprite 107%, for example, it will break pixels everywhere and it will be a mess, but if you scale it 200% each pixel will now be 2 pixels wide and tall, so it will look nice and sharp.

A Bigger Canvas

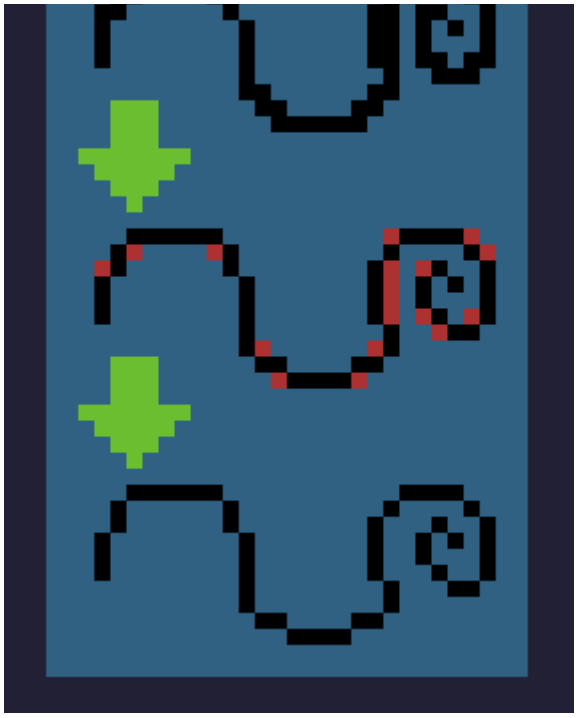
Now that you got the basics, like creating a new file, saving and drawing into the canvas, let's try drawing on a slightly bigger canvas, **32 by 32** pixels. We'll also use a bigger palette now, try the **AAP-Micro12** (by AdigunPolack). This time we will draw a shovel.

Unlike the 16 by 16 sprite, we can actually fit some outlines here, so let's start with that. Here's my process breakdown:

Step 1: Lines

Step 1

This line style is what we call a pixel perfect line, it's only 1 pixel wide and it connects diagonally with other pixels. When making lines like that we avoid unintentional edges, like here:



Aseprite also has a really good feature on the brush settings to do that almost automatically: with your **brush tool** selected, click the **Pixel-perfect checkbox**. Just don't forget to toggle it off when not working with outlines because it will probably annoy you.

Aseprite Pixel perfect algorithm

Step 2: Base colors

Step 2

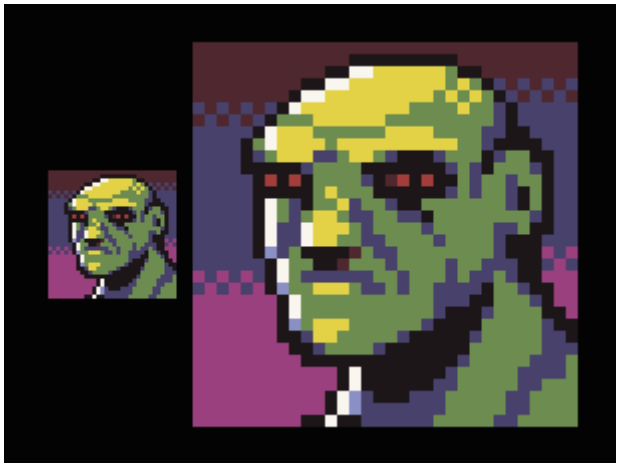
The good thing about having only so few colors to choose from is that you won't be overwhelmed by too many options. That's why it's much harder to work with a lot of colors, if you have a color in your palette there's no excuse not to use it at it's best. Try

to think of it as a puzzle, experiment a lot, even weird or unusual combinations until you find what you believe is the "best match" for each area.

Step 3: Shading

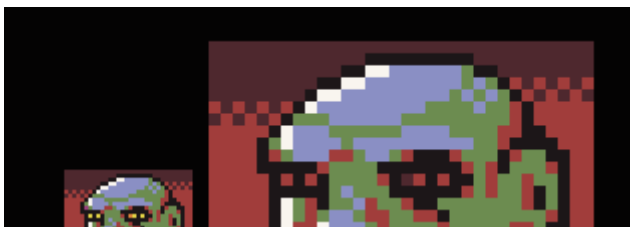
Step 3

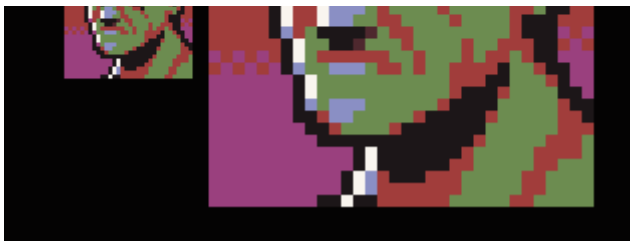
Use your palette to make light and shadow in creative ways. Since you are working with a very restricted palette, you won't have every hue with different brightness, so you will have to improvise.



Improvising shade tones with different hues

In the example on the left I'm using the same palette you are, the **AAP-Mini12**. When I drew this green dude I didn't have any light green color, so I went with the nearest hue I had available, which was yellow. The same thing happened with the shadow, I chose blue because it was the closest dark one. But what if I went the other way? I could get a brighter blue and darker red, right? Well, not really:





Shade tones using inverted hue

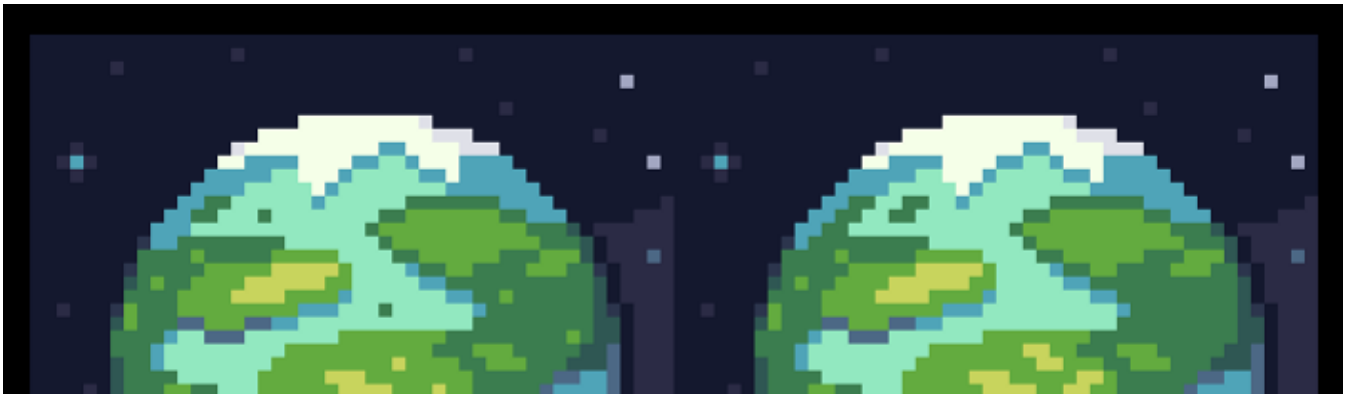
It's a cool effect, but clearly there's something wrong. Usually you will want the cold hues to be your shadows and warm hues to be your key light, or they might look weird. This is not a stone-written-rule or anything, there are many exceptions, but when not sure, just go with it.

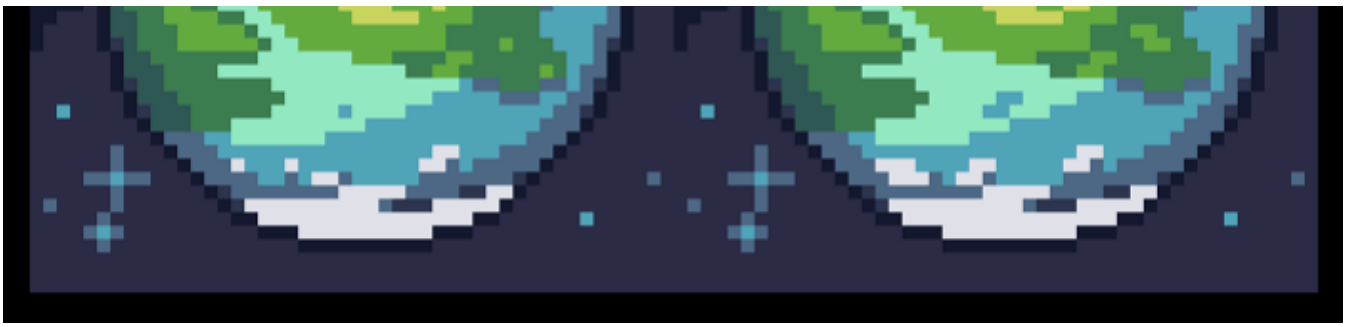
Step 4: Anti-alias and polish

Step 4

This is the part of the drawing where you try to make the pixels a little less "pointy". Manual anti alias is a complex subject, and we probably will need a whole article to discuss just that, but the theory is, you will use mid tones to simulate "half pixels" and soften the edges. But don't worry too much about this yet, for now focus on making your sprite as readable as possible.

Another good idea in this step is to hunt down some orphan pixels to reduce noise. Orphan pixels are pixels that are not part of a bigger group of pixels of the same color and are not part of the anti-alias, like this:





Removing orphan pixels

You see the little 1-pixel-islands on the left? Those are orphan pixels, as you can see the planet looks much better after we merge those pixels with some other nearby pixels of the same color.

And what about the stars in that example? Well, they are there to prove that orphan pixels are not always bad, those stars work exactly as intended, creating a noise texture and bringing up the contrast in the background.

The idea is not to mindlessly remove orphan pixels, but to through them and ask yourself: does this pixel really need to be alone?

Now What?

Now it's time for you to experiment with more colors and bigger resolutions! But go slowly, maybe 48 by 48 and 16 colors and so on. If you are really starting out I would avoid animation for now and focus on getting comfortable with static images first.

I selected some other pixel art guides that I really like if you want to do some research:

- Pixel art tutorial by Cure
- Pixel art tutorial by Derek Yu
- Pixel art tutorial by Arne

I also make some tutorials about specific topics or aspects of pixel art and game design, you can see them all here:

- My Patreon page
- A compiled list of all my tutorials

Keep reading the part 2 here!

How to start making pixel art #2

Cluster sketching and painting

Pedro Medeiros Follow
Aug 4, 2018 · 6 min read

This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

Also, this is the part 2 of a series of articles, read the whole series here in the Pixel Grimoire.
I'm using Aseprite for this tutorial. In this article I'll teach a technique for sketching and drawing pixel art that is similar to the process of a traditional painting. I usually call this technique *cluster sketching*, since I start with big color clusters and refine them until I'm happy with the result.

What is a cluster

Some pixel clusters highlighted

A **cluster**, also called **color cluster** or **pixel cluster**, is a continuous group of pixels of the exact same color. There's some debate whether they can connect diagonally or not. I believe they do connect, I call that a *weak connection* and I try to avoid them, but I don't worry too much about it.

While making pixel art, my focus is to have **as few** clusters as I can and to avoid one-pixel clusters by all means. These one pixel clusters are also called **orphan pixels** and they usually are responsible for the image looking noisy and confusing.

Identifying and changing orphan pixels to small clusters

Sometimes you can just remove the orphan pixel and sometimes you need that detail. For the latter I have some favorite shapes to replace the orphan pixel, they are the green shapes below the cookies. But there are also some cases where they can be used, for texture, anti-alias (I'll talk more about that in future chapters) and for strong details, like the red eyes of this skull:

Orphan pixels are not always bad!

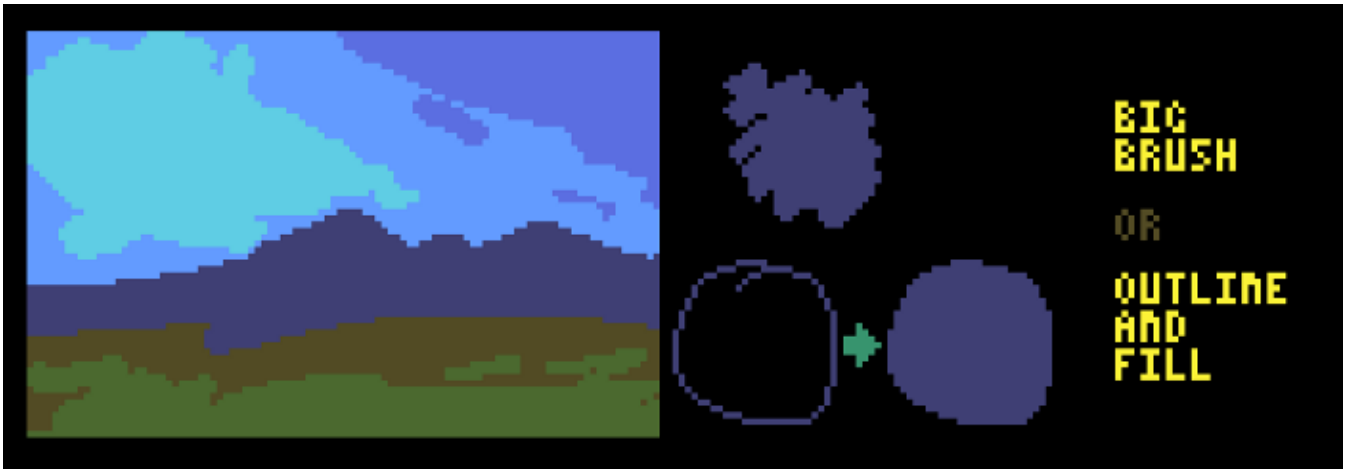
Let's Draw

Let's draw something! I'm making a little scene using the **DB32** palette and a really big resolution, 100 by 64 pixels. For this exercise a drawing tablet is highly recommended because making natural strokes helps a lot with the result.

If you think this scene is too complex for you, feel free to tone it down by removing some stuff, like the building or the person. Always draw something comfortable when trying out a new technique. If you feel like the canvas is too small or too big you can also change the size a little, but I wouldn't go smaller than 64 or bigger than 128 pixels, at least not for now.

Step 1: Big clusters

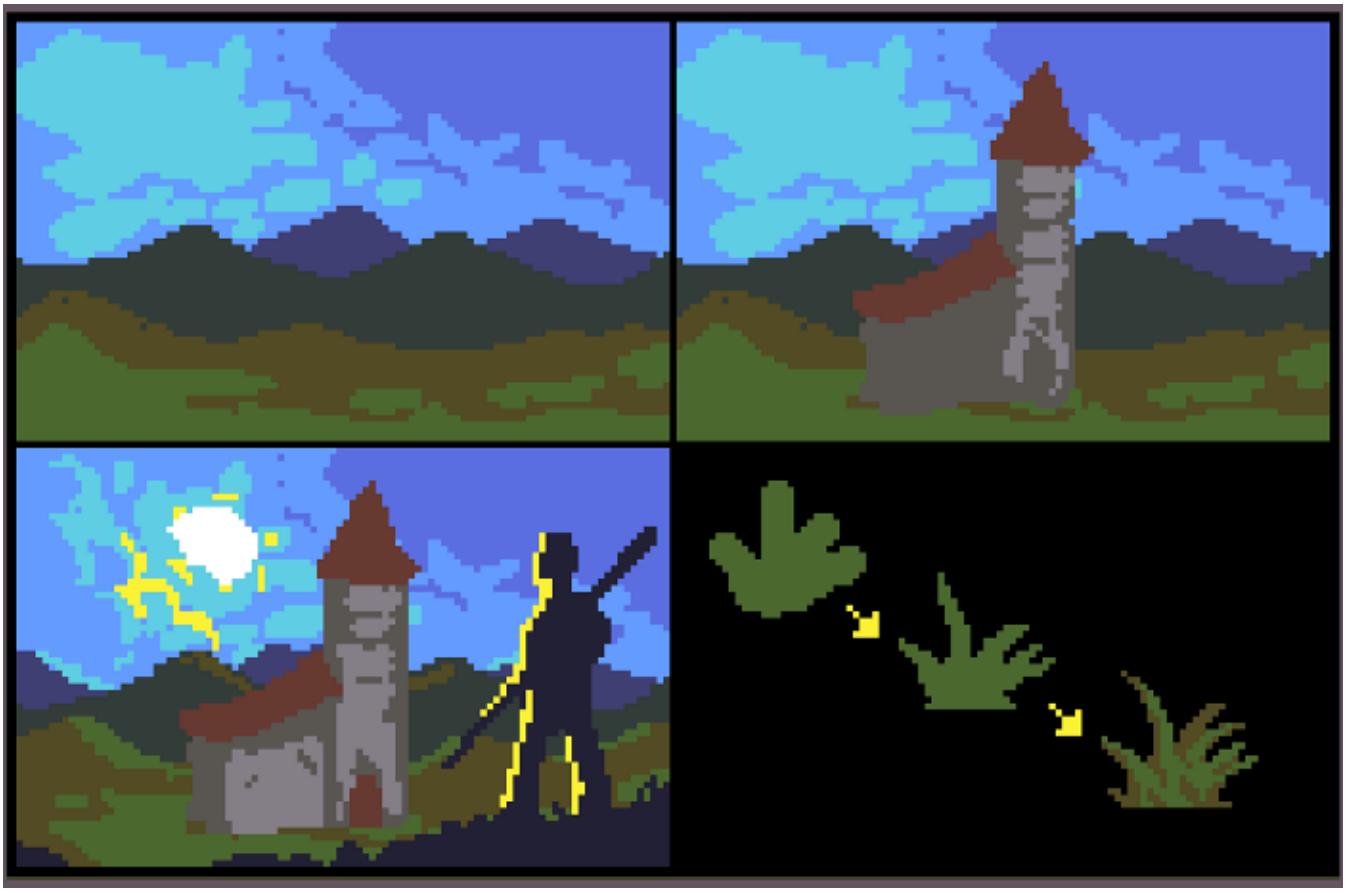
Last time we sketched with lines and then we filled the interior with colors, this time we will start straight away with the colors. Just do a very messy version of the image you want to do, focusing on choosing the colors and the vibe you want. Favor gestural movement and do **not** add detail.



Step 1

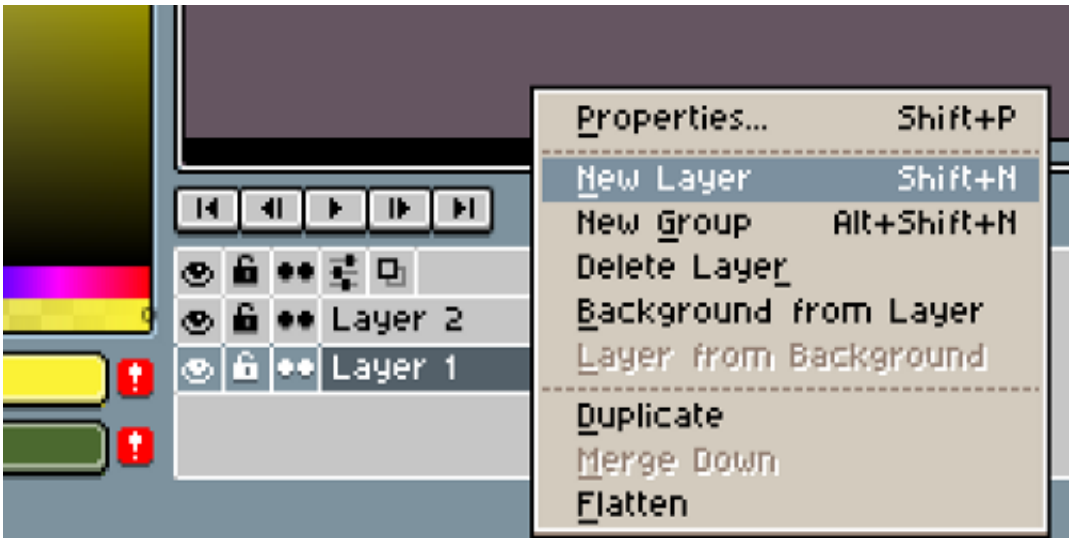
The idea with this technique is to start with big blobs of color and go smaller each step. This is one of the few situations that it's OK to use the brush tool with 2 or even 3 pixels size. To increase and decrease your brush size, use the "+" and the "-" keys. Another way to do this is to draw the outline of the cluster you want and then just fill it with the paint bucket tool (G key is the shortcut).

Step 2: Refine



Step 2

You can also see that I start with the far back, like the sky and mountains, and then add the building, and then the silhouette in the first plane. This is a common painting technique and like most painting techniques it also works well with pixel art. The idea is to have your foundation ready before placing things on top of them, this way it's easier to choose colors and the scale of the objects.



I also like to work with layers on images like this, with the sky, the building and the first plane being in different layers. To create a layer, right click the "Layer 1" in the

timeline on the bottom of the screen, and select **New Layer**. You can have as many layers as you want but I always try to work with as few layers as I can, otherwise things can get too messy.

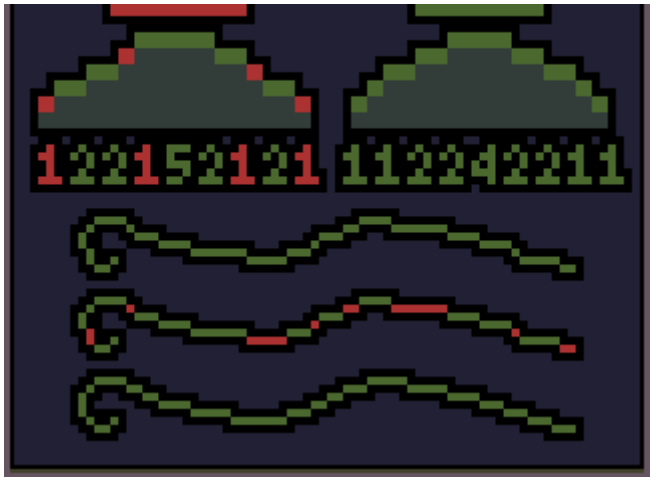
Step 3: Fix jaggies and add detail

Pushing pixels around to fix jaggies

What are jaggies? OK, this will sound like it's super complicated, but I promise it isn't.

Jaggies are unintentional corners that appear in pixel art, usually when making natural hand movements, as a side effect of the lack of anti-alias.

Imagine that the border or line you are trying to fix is a staircase, you need to apply some logic to the number of pixels in each step. We need to manually count pixels and make sure that the number of pixels in each "step" increases as the curve approaches a horizontal angle and decreases as it approaches a vertical angle.



Identifying and fixing jaggies

Let's break this down. Every line or cluster border should follow some kind of logic. You can see in my example that the amount of pixels in each step on a *perfect curve* goes up and then goes down. That's how curves behave, they usually follow some geometrical progression, exponentially going up or down.

The **jaggies** are just "missteps" on that logic. It's when a step size suddenly goes down and goes up again in the middle of a a curve (or the other way around). The numbers can go high or low really fast, that's not a problem, as long as the curve logic applies.

To fix them you push pixels around to make the steps follow a steady number or to make it increase or decrease correctly.

Let's get back to our image now.

Final step

The idea here is to "draw the rest of the owl" mainly by looking for jaggies and fixing one by one. While doing that I'm constantly adding details, stronger contrast, better

light, etc. It's not a very organized process but it gets the job done.

Now What?

Now it's time to practice! If a complete scene is too much for you, try starting with something simpler, like a **rock** or a **tree stump**. This technique produces very organic and painterly results, so it's great for backgrounds and for drawing nature, like plants, water and mountains.

Keep reading the part 3 here!

Thanks to Amora B..

Design Pixel Art Tutorial

How to start making pixel art #3

A basic Aseprite animation

Pedro Medeiros Follow
Aug 28, 2018 · 6 min read

This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

Also, this is the part 3 of a series of articles, read the whole series here in the Pixel Grimoire.

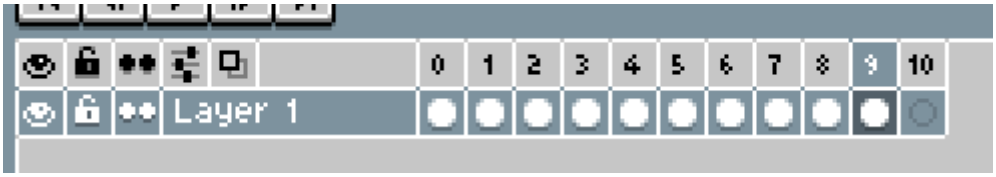
What is an animation?

The breakdown of an animation

An animation is an illusion of movement caused by a sequence of images played in a specific order that shows progressive phases of that motion. Our job as animators is to make that sequence look as convincing as possible.

The timeline

The fist thing we need to understand is the timeline. It's a way to represent multiple images in a single file. Each column is a complete image and has a number assigned to it, we call it a **frame**.



The Aseprite timeline. You can see that the frame 9 is selected and 10 is empty.

The easiest way to create a new frame is to press ALT+B. This will make an empty frame right in front of your currently selected frame, and select it. You can change the selected frame by clicking on it or by pressing the ‘ , ’ and ‘ . ’ keys (look for the ‘<’ and ‘>’ symbols, it’s easier to remember like this.

The preview window

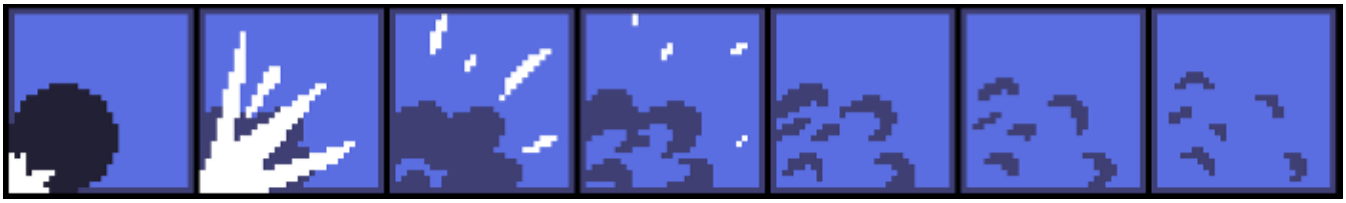
Experiment by drawing some colors on multiple frames and hitting play (*Enter*). You can also preview your animations using the preview window that can be toggled by pressing *F7*.

You are probably thinking “There’s no way I’ll remember all these shortcuts” and that’s OK. Memorizing shortcuts takes some time and muscle memory, but know that mostly everything can be done using only the mouse. When you can’t remember a shortcut look for that command in the menus or in the Aseprite official quick reference.

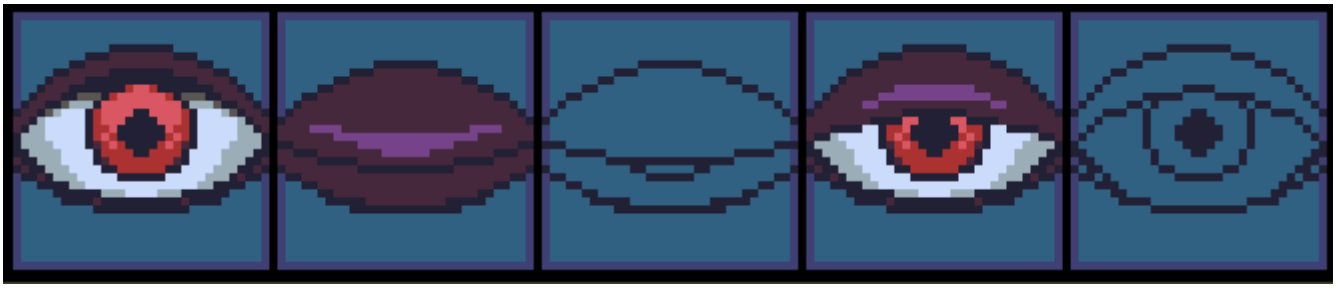
A very simple animation

There are many techniques on how to animate, the order you should draw the frames and how to optimize everything, but for now I’ll try to explain the simplest technique I know: a straight-ahead animation of a bouncing ball.
“Straight-ahead” means we’ll draw one frame directly after the other, as opposed to

drawing all the important poses of the animation first and then the middle ones. Start with a 32x32 file with a palette of your choosing.

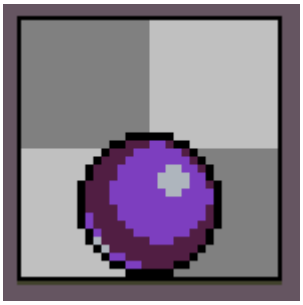


An example of a straight-ahead animation



In a pose-to-pose animation you draw the key frames first and then fill the gaps

The first frame for this animation I call the "*Still*". It serves both as a concept art, to define the style of your animation, and as the resting position of this sequence.



Frame 1 — Still

You need to pay attention to the amount of details you'll be adding to this frame, because the next ones will be following the same style.

After this is done we can duplicate this frame (ALT+N) and just move it upwards 4 pixels. Like this:

Frame 2

And now for the next frames, make it go up 3 pixels, then 2, 1 and let's hold it in place for one frame before reversing it.

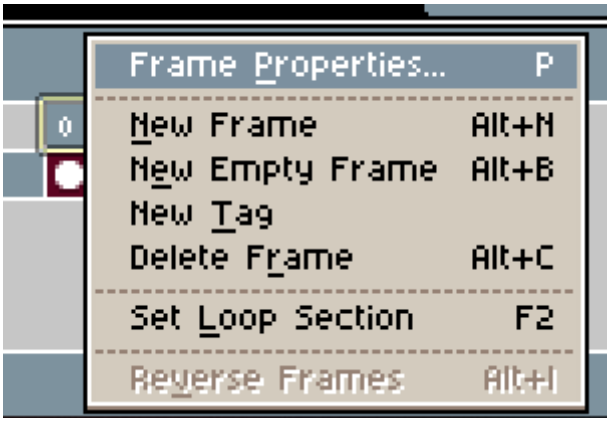
All frames

We made a little bouncing animation! Well, it's not very good yet, but we're going to improve it.

A very simple bouncing animation

Timing

While cleverly using timing is a complex subject and could even have a whole tutorial dedicated to it, I'll focus now on the simple technical aspect of it and let you experiment.



Accessing frame options

So let's add some timing. The ball stops when it hits the ground, waits for a little while and then magically jumps up again. To increase the duration of a frame, **right click** on a frame's number and choose *Frame properties*. Then you can type how long you want the frame to last. Let's try 300 milliseconds for this.

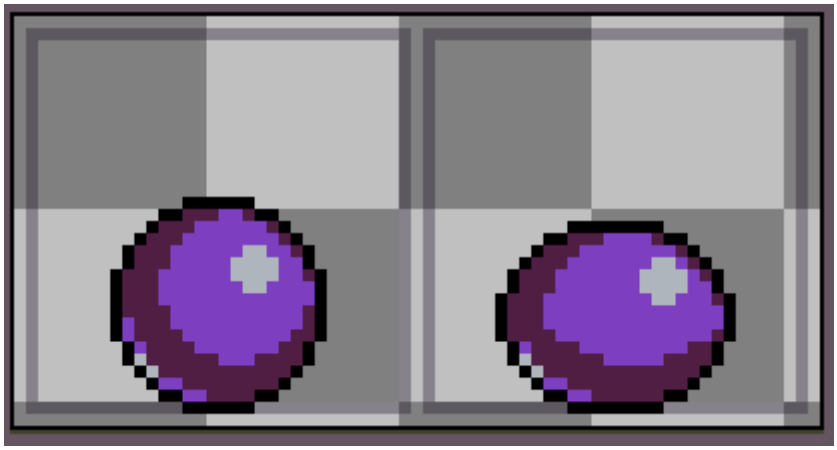
Remember you can also select multiple frames to change their duration at the same time. This can be especially useful to speed up or slow down a whole animation at once.

Squash and Stretch

Here's my old tutorial about squash and stretch, it can be used as a quick reference if you need it

A simple thing we can do here to improve this animation is to add some squash and stretch, a really cool technique to make the movement seem more fluid and natural. It consists in elongating or flattening the moving object in the direction of the movement while keeping the volume.

Let's duplicate the first frame and squash it horizontally while flattening it a little vertically, so we can keep the total volume. Keeping the volume is very important so your object doesn't look like it got smaller or bigger. Of course that's a rule we learn to break as we get more experienced, but for now let's stick to it. This is also called an "anticipation frame" and it's mostly used to make the movement happening on the next frames look more intense.



Squashing to anticipate the jump

Now let's duplicate and change the frame right when the ball hits the ground in a similar way, maybe making it even more exaggerated.

Contact with the ground

The last thing is to stretch vertically (always flattening horizontally) the first frame of the jump and the last frame of the fall, when the ball is at their fastest. And we are done! Let's see our result:

An improved bouncing animation

A bouncing loop, made by removing the still and anticipation frames

Saving the animation

While saving the file in the *.aseprite* format will preserve the animation, you will probably want to export your animation to post it online or to use it in your game.

Online, the easiest way is to save your image as *.gif*, using the *File>Export…* command. Just check the *Export for twitter* if you want to change the last frame's duration to 1/4 of the duration so it loops perfectly even after Twitter converts it to MP4.

When exporting for games you will usually want to save as *.png*, as usual, but the animation will have to be broken down in a sprite sheet or image sequence. To save it as an image sequence, simply export the file to *.png* format with a number in the end of

the file name, like “bounce00.png” for example. This will create multiple files, like “bounce01.png”, “bounce02.png” and so on.

Some game engines will need the file in the sprite sheet format. You don’t need to do that manually, just check *File>Export Sprite Sheet* and play around with the settings, you can change a lot of parameters there.

Now what?

Try experimenting with simple movements!

Next, I would recommend experimenting more with the timeline and to try making other animation tests. You can look on my tutorial gallery and select a tutorial of your interest and try to mimic it.

Another idea is to try the classic beginner’s exercise of making an animated flour sack walking and jumping around. Just keep the resolution and color count low for now, so things don’t get overcomplicated.

Keep reading the part 4 here!

Design Pixel Art Animation Tutorial

How to start making pixel art #4

Basic Shading

Pedro Medeiros Follow
Nov 22, 2018 · 7 min read

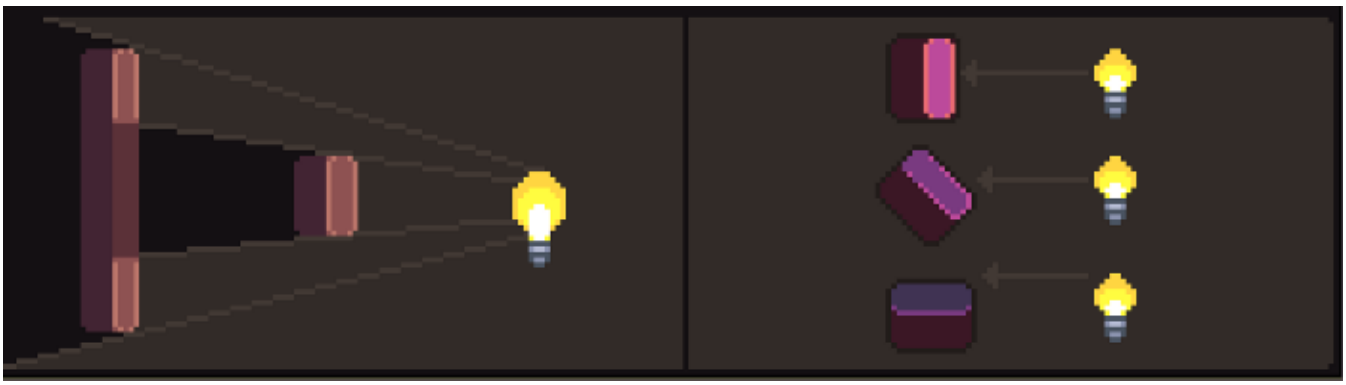
This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

Also, this is the part 4 of a series of articles, read the whole series here in the Pixel Grimoire.

This article is a little longer and text-heavy than the previous ones, so buckle up!

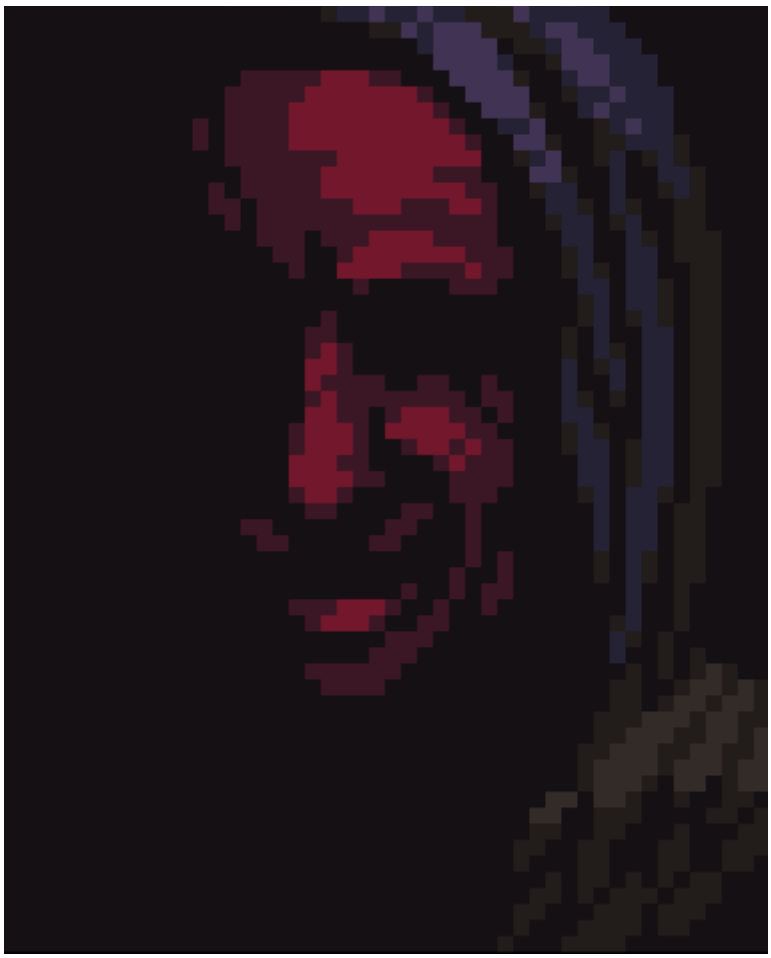
How light works

We see things because of light, and when we draw something we are actually trying to represent how the light reacts to that object. Even if you haven't studied this before, you already know how light works, you see it every day: it bounces off some materials and occasionally refracts to others.



Light blocking and perpendicular light

One less obvious aspect is that the angle at which the light hits the object also matters, a perpendicular light is darker than a direct light.



You need to try to simulate that light behavior in your head, almost like you are a human rendering machine. Since that's very hard and prone to errors, a good way to study how light works if to use photo references. Photo references are a great thing, you don't need to necessarily copy the photo, but you can use it to understand how the light behaves in that particular case. Using a photo reference is always a good practice and there's no merit in not using one.

Basic structure

When shading an image it's good to have a structure to follow, I'll make a basic glossary with a brief definition of each term.

Volume shadow: The most common type of shadow, it's a self-projected soft shadow. It's the result of the light being blocked by the object's own volume.

Terminator: It's the transition zone between the light area and the dark area of an object. It can be soft or sharp. In pixel art we favor sharp transitions to avoid *banding* (more about that in the future).

Projected shadow: When one object projects a patch of shadow into another. This is usually a very sharp shadow.

Reflection highlight: Also known as *specular highlight*, it's the brightest spot in the object. Glossy and reflective objects have small and focused highlights. Rougher objects may not have a reflection highlight.

Highlight: The basic lighter area of an object, imagine it as the reverse of the volume shadow.

Rim Light: When the light is coming from the back it looks almost like a bright outline. This is usually cast by a secondary, dimmer light.

Bounce Light: It's a little hard to see sometimes but this is a slightly brighter spot in the volume shadow, caused by the light bouncing off the ground and back onto the object.

By no means you need to memorize this, but if you are not sure on what to do on your painting, check this list. Another thing to keep in mind is that all of this will not even fit

in most pixel art pieces, especially the lower resolution or the lower color count ones. As always, remember: good pixel art eliminates all unnecessary detail.

Working with a photo reference

Let's do an exercise here, let's try drawing this photo:

Our reference photo

First, let's start a new file, I'll use a 48x48 canvas and I'll use the AAP-64 palette. I encourage you to take a picture of something simple and do the same, or just use this one. Try to follow my steps.

Like in my previous tutorial, I'll work with big clusters of color, trying to emulate how the light behaves in the photo. Here's my whole process, and a resized version of the photo:

As you can see I took a lot of liberties in my design, I changed the pose a little, removed the specular highlight and made the Daruma a little more like an egg. The main reason here is that we don't want the drawing to look like a resized photo, we need it to look better, so I made adjustments so it would **be more readable** and **look more interesting**.

I focused on the details that I think are important. Since we're working with such a low resolution, some details must not be included while some others need to be highlighted. Look at the resized mouth, it's barely there, I made it bigger and more noticeable, because it'a an important detail. Same with the eyes, they are now bigger and regular.

Let's go step by step:

1. Start with the basic colors and shapes, I like to start with the darker colors and paint the brighter ones later, but that's a personal preference.
2. Paint the basic light/shadow inside the object, keep your color count low for now.
3. Draw projected shadows.
4. Draw the details, engravings and other small parts.
5. Correct the shapes, reinforce the shadows and highlights.
6. Finish up with some extra anti-alias and outlines, if necessary.

If you're still not happy with your drawing, don't worry, go at your own pace and keep practicing. Try drawing something simpler and focus on eliminating unnecessary details.

Identifying Faces

On our next drawing, let's try something different. You will work on shading this image:

Tiny slope

Again we will use the AAP-64 palette and a 48x48 canvas. The idea here is to illuminate the scene properly. The main difference of this and the other image is that this one has mostly flat *faces* and the other is round.

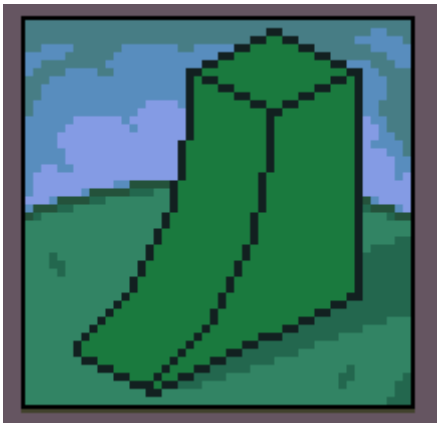
Flat faces usually have a **uniform color** in their entirety while rounded shapes can have color ramps. Flat faces can have ramps if the color is not very strong or if the object is too close to the light source, but I would avoid that when possible, or keep the variation to a minimum.

With that in mind, choose a light direction and paint our sprite. Then come back here to check how I solved this and see some common mistakes.

I didn't put a light source in the sprite because this is something you need to practice too, choosing a good light source is a skill as important as rendering the scene correctly.

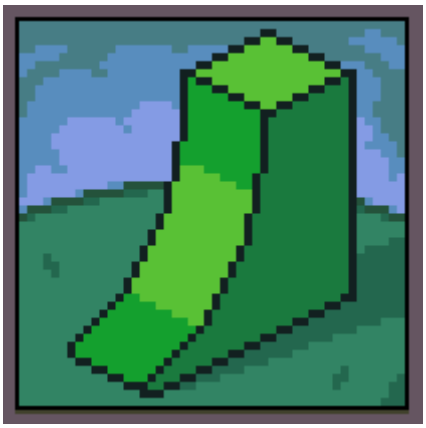
My version

The first thing I did was the background and the projected shadow, this will help me set the tone for the whole image.



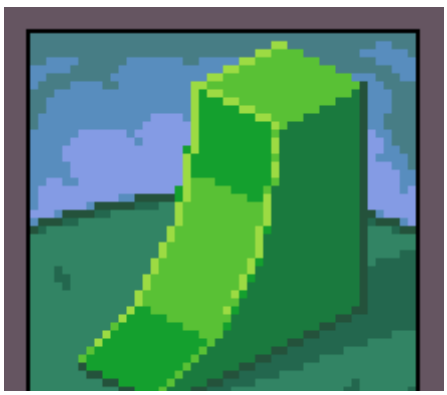
Step 1

After that, I tried to set the color for each face: the top and side faces should have uniform colors, the slope can have a gradient. I only used two colors because I want to keep it simple and the banding effect (more on that on part 5) to a minimum.



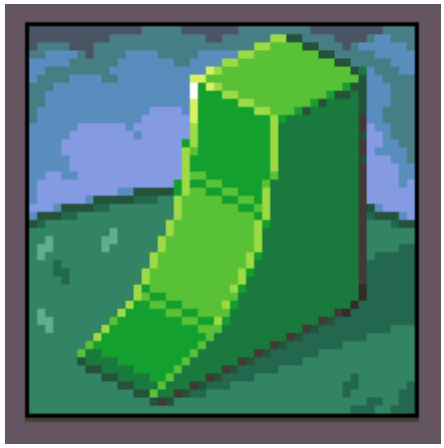
Step 2

The next thing are those outlines, they were bothering me a lot. So I tried to imagine them as other tiny faces, softening the shape a little.



Step 3

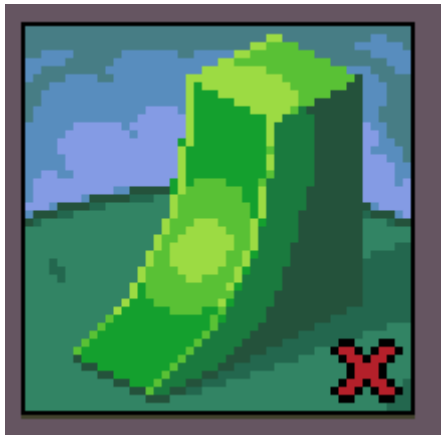
Now the contrast was a little too low, so I added a small specular highlight and deepened some of the shadows. I also added some cheap dither to make the slope transition a little better.



Final step!

Common Mistakes

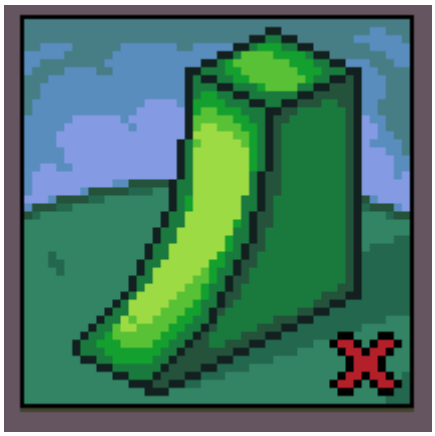
Here's some of the common mistakes when shading an object, so try to identify if you are doing any of these things. Also note that the examples here are exaggerated, and this can happen in smaller amounts in your drawing.



Soft faces

Soft faces

This usually happens when trying to soften the light on the object a little too much or in the wrong direction. Flat areas should have mostly uniform colors and curved areas should only change the color in the direction of the ramp.



Pillow shading

Pillow Shading

This is a famous one, it happens when trying to shade an object with no clear light source and making a generic shadow around the shape.
This is best avoided by having a clearer light source and not darkening the edges of a face.

Flat light

Flat Light

Another common mistake is to ignore the actual shape of the image. Remember that your object is a representation of a 3D object. If you're not sure of how light should behave, try researching or even making a photo reference.

Now What?

Now it's time for practice! Grab more photos and other illustrations and just straight out copy them, the more the better. Also try experimenting, use a very limited palette, can you represent a color using another? Study here is key, get zoomed-in images, see how their pixels look like, re-scale pictures and see what works and what doesn't.

Don’t forget to look around you. When walking on the street, pay attention at how the light shines on buildings, how it shows through your fingers. Nothing will teach you more about light and shadow than exploring the details you see everyday.

Good luck!

Keep reading the part 5 here!

Design Pixel Art Game Development

How to start making pixel art #5

Anti-Alias and Banding

Pedro Medeiros Follow
Nov 7, 2018 · 6 min read

This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

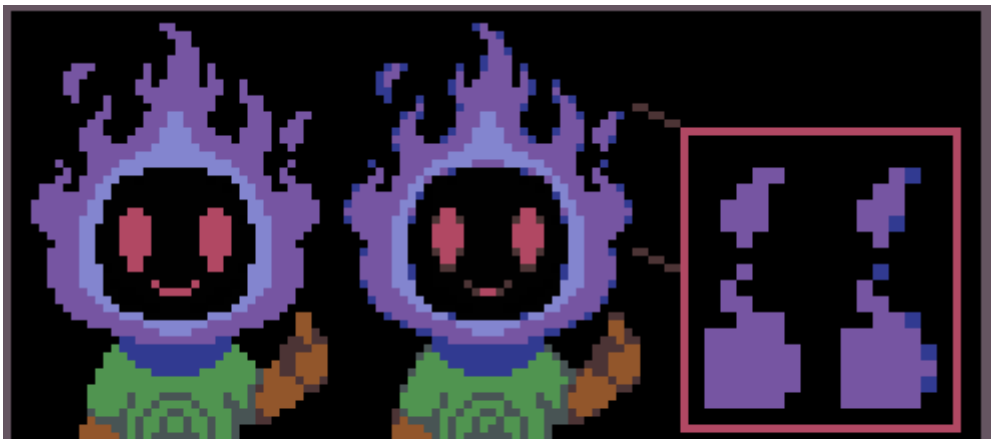
Also, this is the part 5 of a series of articles, read the whole series here in the Pixel Grimoire.

This is a slightly more complex theme, so the article will be a little longer and more advanced than the others. Don't worry if you don't understand it all at first, a lot of it is very subjective and opinions about anti-alias and banding differ vastly in the pixel art community.

I won't go into software at first but focus on exploring and explaining these concepts. In the **Now What?** chapter I'll recommend some exercises.

What is alias and why people don't like it

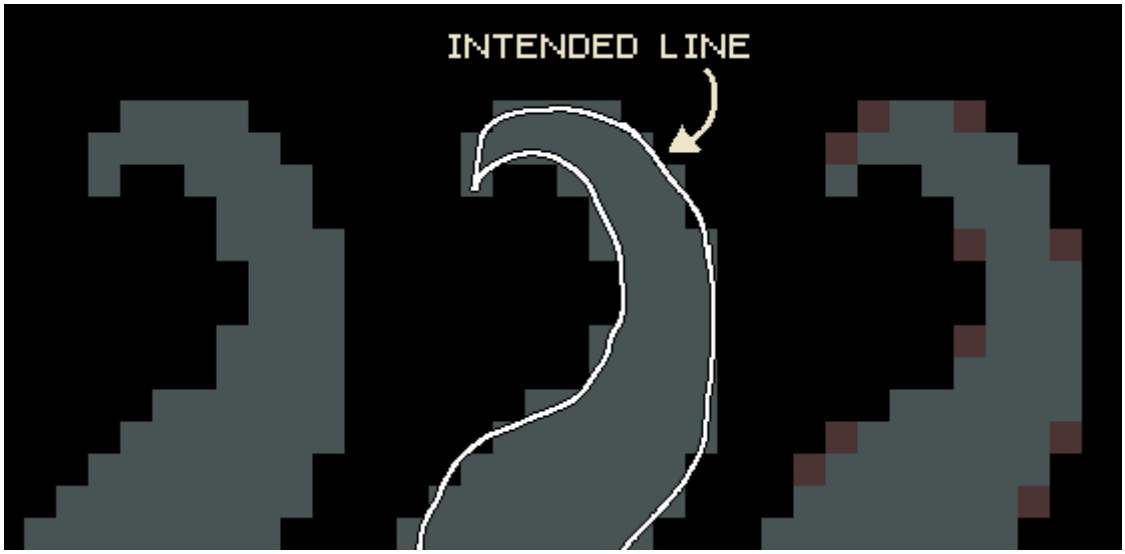
When we say *alias i*n pixel art we are usually talking about *jaggies*, or the *staircase effect*. An effect that's mostly noticeable in lower resolutions. This effect is not necessarily bad by itself, it can even be used on purpose depending on the art style you are aiming for, but sometimes it can cause the image to become very hard to read.



An aliased and anti-aliased illustration

As you can see in the example above, the first image has some very sharp edges, especially around the fire. In some areas it's even hard to understand what's supposed to be going on in there.

This happens because pixel art is a low resolution representation of something. If you think like this, it's possible to separate an "intended line" from the actual pixels.

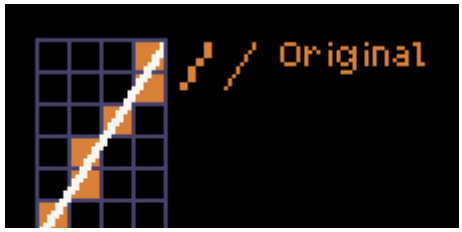


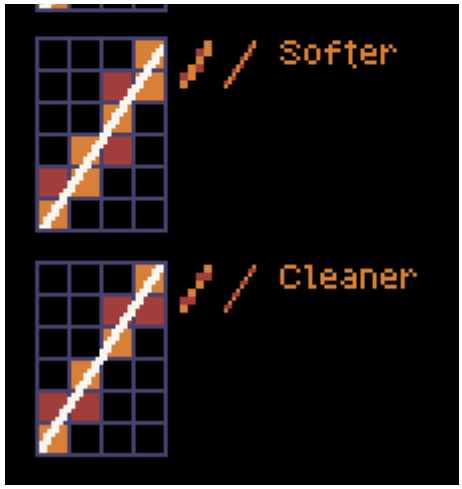
Line intention versus pixel art result

Basically the whole pixel art process consists in translating that intended line into the lower resolution of the canvas and **deciding which pixels to fill**. Some pixels are obviously inside the our lines so they get filled, some are clearly outside so they don't. The problem is that some pixels just barely touch the line or are half filled, those are the problematic ones. Anti-alias is the process of using a color in between to fill some of those pixels to create the illusion of a softer transition.

How can we make anti-alias

Quick recap: sometimes we need to represent a line or curve that doesn't fit in our pixel grid, so we use semi-tones to make them look better.





Different solutions for the same problem

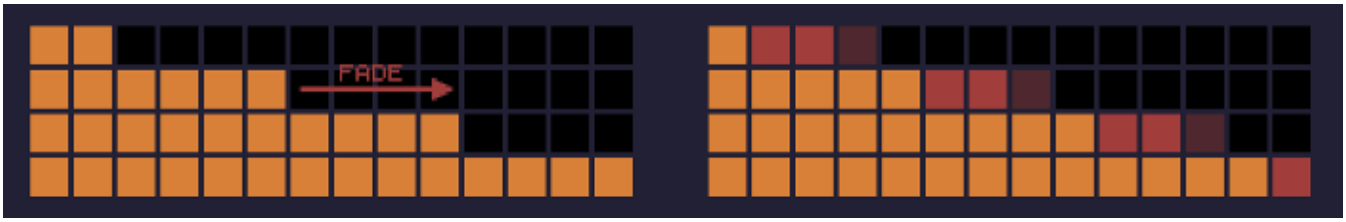
There's no single solution for this problem, you can solve your "intended line" in many ways, depending on your resolution, art style and how many colors you have available.

The theory is that a pixel with a mixed color halfway counts as a half pixel when zoomed out. You can use those to try to round those pixel edges a little.

A good method for making anti-alias is to search for any *staircase pattern* that's longer than one by one pixels:

Simple line patterns

After identifying the pattern with steps two pixels or longer you can start to soften them, by adding a half tone in after the step ends. The amount of pixels you should fill depends mostly on the length of the step and the available colors. Sometimes this can make the object expand too much, to compensate for this you can use the half tone to "eat" one (sometimes more) pixels from the actual original shape.



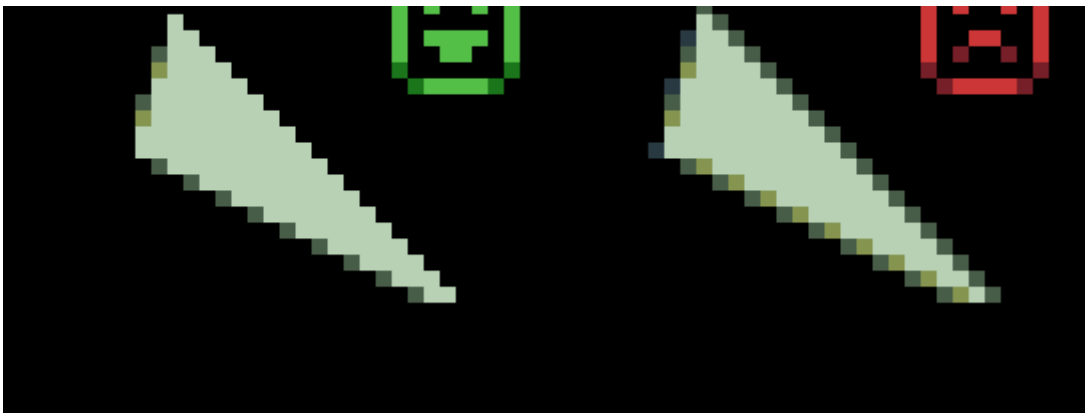
Anti alias on a wide slope, expanding both forward and one pixel back

Note that changing the color hue or saturation doesn't matter too much, as long as you are interpolating the color values correctly. This is specially useful if you are working with a very small palette.

Improvising half-tones with very different hues

This is already complicated when working on simple slopes, and when working on real drawings it gets exponentially more complex. On a small canvas, anti-aliasing one object can affect another in weird ways and create a lot of side effects. Let's talk about some common ones.

Too much anti-alias



Everything is blurry!

The most common mistake is when the anti-alias is overdone causing a blurry effect. Avoid this by using fewer color halftones, fewer steps and do not use anti-alias in 45 degree lines (one by one steps) or in straight lines.

Anti-alias is too short

Not enough steps or too short

This is less common and can be done on purpose, specially on limited palettes, but sometimes the anti-alias halftones are too short in comparison to the length of the step. Always remember that a long step should have a proportionally long anti-alias halftone strip.

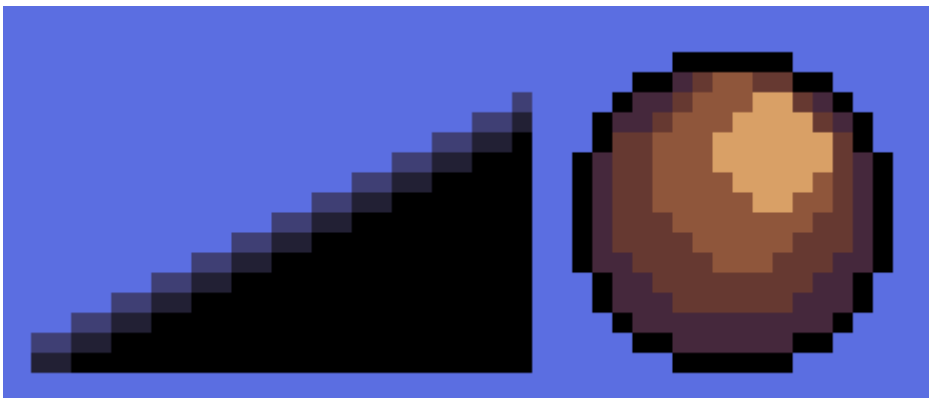
Wrong values on the halftone

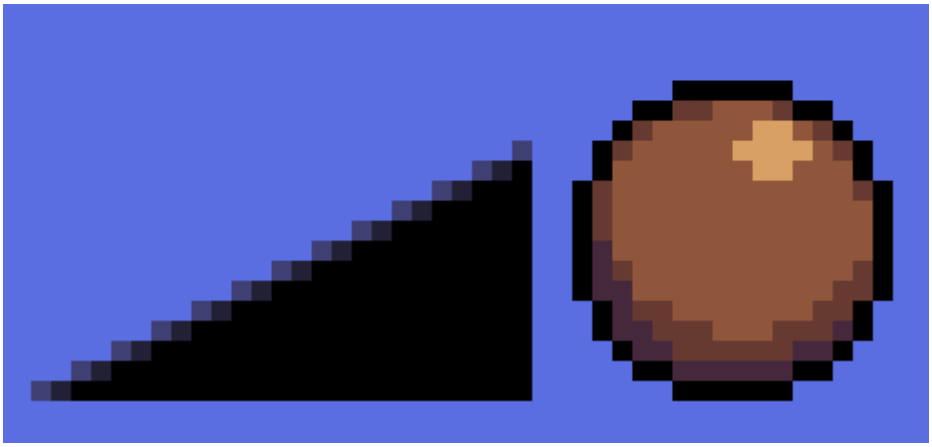
The first image has a reversed value ramp

I mentioned before that the halftone can have different hues as long as the value feels like a bridge between the two colors you are mixing. The problem here happens when that's not taken into consideration, making the halftone too dark or bright in comparison with the background.

Sometimes, when the background is lighter than the object, this can be used to emulate a shadow, which can work. Just remember that in that case you are not making anti-alias but just shading an object.

Banding





Some really bad banding and compressed bands

Banding happens when we have multiple color bands that are clearly distinct from one another. This is usually very problematic, making the object look flat and sometimes blurry.

My old shading tutorial

In the sphere example you can see that I compressed the bands into tiny areas. I call that technique, obviously, **band compression,** and I mention it in my shading tutorial.

In the slope example, though, there's no way you can compress those bands even more. So what I did was rotating the direction of the gradients. This will also create a nice anti-alias, which was initially made in the wrong direction, thus, causing the banding. Always pay attention to that, an horizontal slope should have an horizontal anti-alias and a vertical slope should have a vertical anti-alias.

Now what?

Now I would recommend you actually trying out the things I explained here. You can start with a simple 2 by 1 black slope, and then try a circle. Always keep your color count low and your file size no more than 48x48.

After they are done you can try some colors and some actual drawing, I would start with something simple, like a sphere or an apple before moving on to more complex shapes, like people.

Always look for banding in your images after they are done or points where your anti-alias could be improved. Remember that each pixel is a decision and each pixel should **improve** your drawing, making it more readable. If the anti-alias is doing more harm than good, just remove it.

Keep reading the part 6 here!

Pixel Art Game Development

How to start making pixel art #6

Basic Color Theory

Pedro Medeiros Follow
Jan 9, 2019 · 6 min read

This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

Also, this is the part 6 of a series of articles, read the whole series here in the Pixel Grimoire.

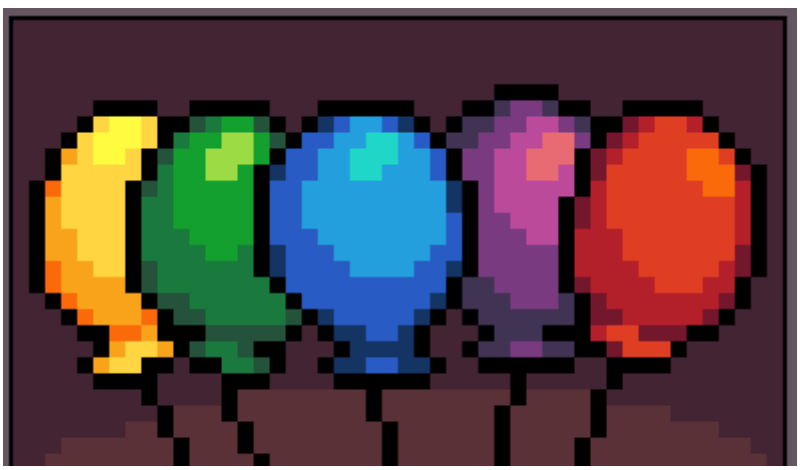
Understanding Colors

Even when using a pre-created palette we still need to think about the colors we are using. My main objective when it comes to colors is to do as much as I can with as little as possible. I'll try to explain some characteristics and synergies between colors.

We can break colors down to 3 main aspects: **hue**, **saturation**, and **value**, or **HSV** for short. There are more ways to break down colors, like HSL, RGB, LAB, CMYK and many others, but I chose **HSV** because it's a very simple and direct way of manipulating color when drawing.

Let's explore these aspects one by one.

Hue



Balloons of different hues

Hue is a property that describes… well, the color of the color. It's also called the identity of the color, if it's red, green, blue, but not how bright or intense it is.

It's important to note that some hues, like blue and purple, can appear darker than yellow, even when the luminosity value is the same. Remember that you might need to compensate this depending on the effect you are going for.

Usually we avoid mixing too many different hues in a single image or it might get too busy. I'll explain some more about this in the **Color Schemes** topic.

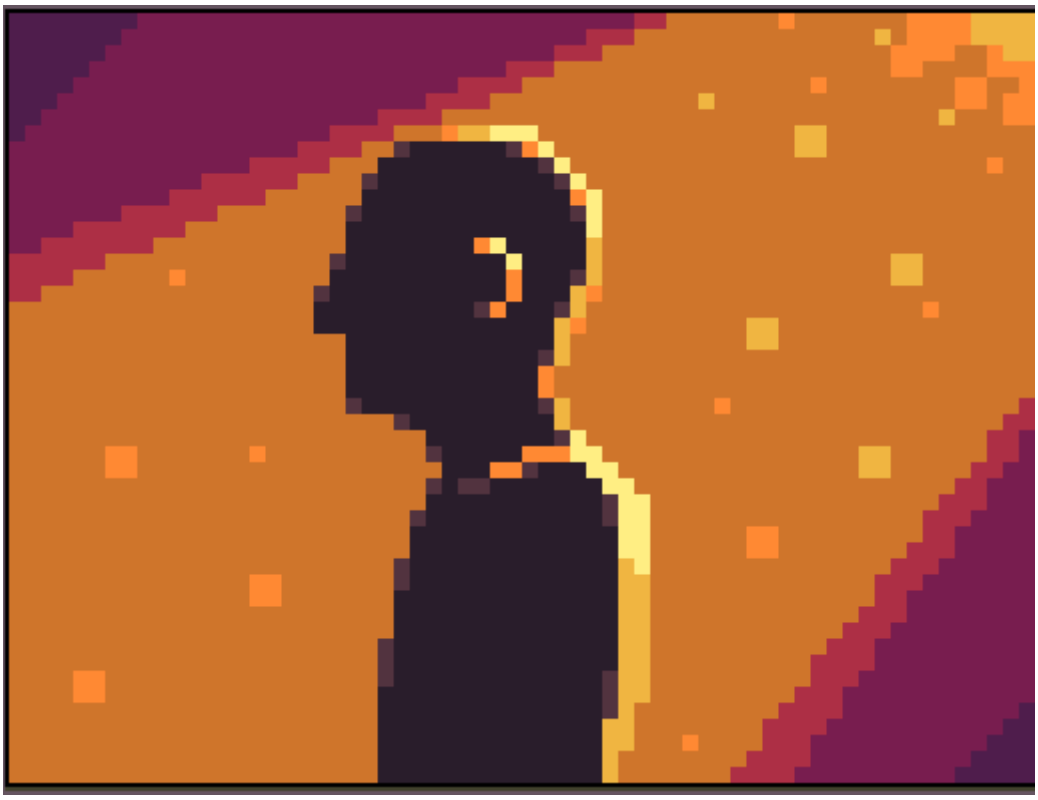
Saturation

Low contrast skin and high contrast details

Saturation is the intensity of color or pigment in a color. A bright red has high saturation, opposing a grey color, which has very low saturation. Usually too much saturation can make your eyes hurt a little, so if you're not sure, avoid using 100% saturation in colors.

Large high saturation areas can also make the eye tired, so again, if you're not sure, try using large low saturation areas with small high saturation details.

Value



High and low values

Simply put, value is the amount of light a color has. A light orange has a high value, a dark orange has a low value. Usually this is directly related to light; where there's light there's a high value color and the opposite in the shadow.

Color Temperature

Color temperature is not part of the **HSV** model but it's a very important characteristic of the color.

Color temperature is dictated by all the characteristics: hue, saturation, and value. In general, you can think something like this:

- Red hues are hot and blue hues are cold

- High saturation is for the extremes (hot or cold) and low saturation are for temperatures in the middle. Pure gray is usually perceived as slightly cold.

- Value is complicated, but most of the time, high values mean hot and low values mean cold.

I like to have images with opposing shadow and light temperatures, usually a hot light and cold shadows, but sometimes the opposite can work too. This doesn't mean that one needs to be red and the other blue, just slightly warmer or colder, it all depends on the ambiance you want to create.

Shading

Choosing colors is especially important when dealing with light and shadow. I explained the basics of shading in my previous article (that ideally you should read before this one). I'll focus now on how to choose colors when shading.

Let's try making a simple drawing of a vase, here's how I started mine:

A simple vase with no shading

Nossa Senhora da Saúde Church (Photo by Heidy K.M.)

When making the light, one might think “I’ll just increase the value on the bright areas!” but that’s not it works in the real world.

Check this picture of a church. You can see that the bright area on the top is clearly more saturated and yellow than the darker area, which almost feels blue in contrast with it (but it’s actually a very low saturated red). Using pictures like this is a great way of understanding color and light.

Let’s try to apply this logic to our little vase. Instead of using a simple value ramp, we will use a hue-shifting and saturation-shifting ramp.

Simple vs. hue/saturation-shifting

As you can see the simple ramp on the left looks a little dull and muddy, while the one on the right is more vivid and interesting. In this case I made the shadow a little more

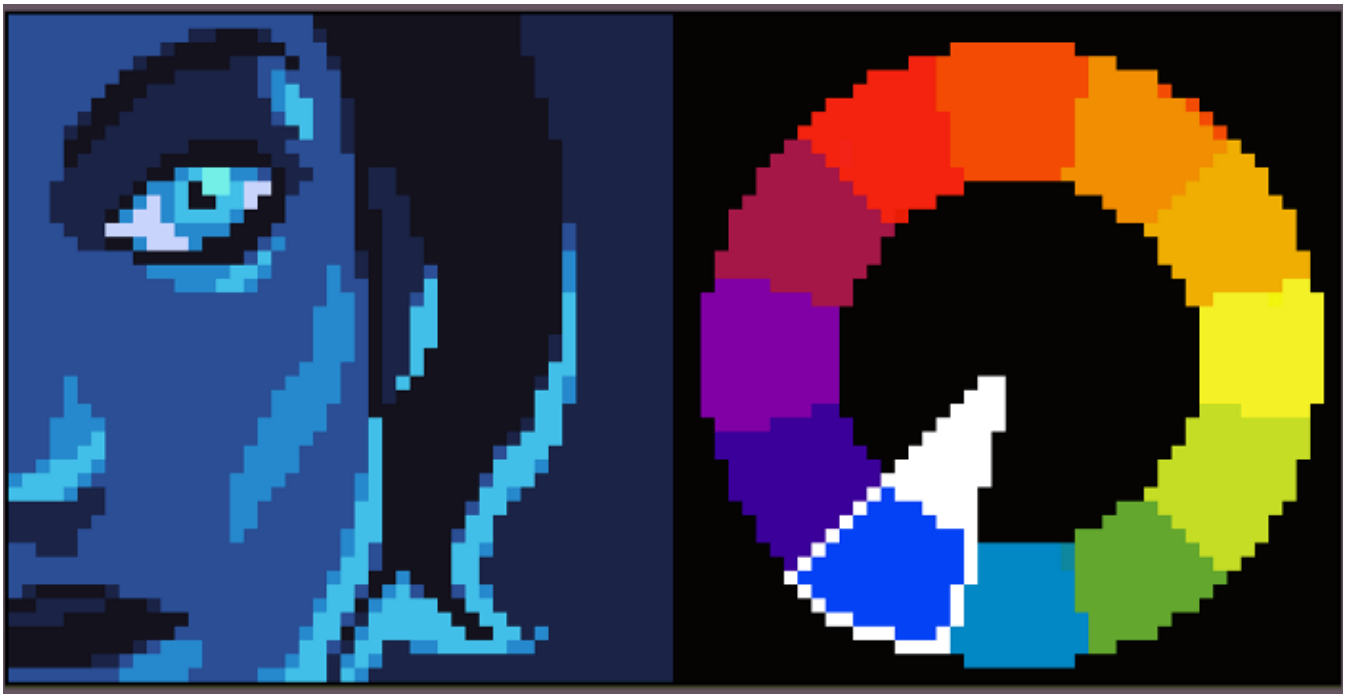
blue and less saturated (colder), and the light a little more yellow and saturated (warmer), almost like the church photo.

When not sure, always look for photo references and experiment a lot! Sometimes just try something crazy like doing a higher value than the mid-tone or using a completely different hue for the highlights, you might stumble upon an interesting technique.

Color Schemes

You can group colors using some clever formations and use that to create your palettes and composition. This is not something absolutely necessary but it's definitely a useful tool for creating interesting contrast and moods.

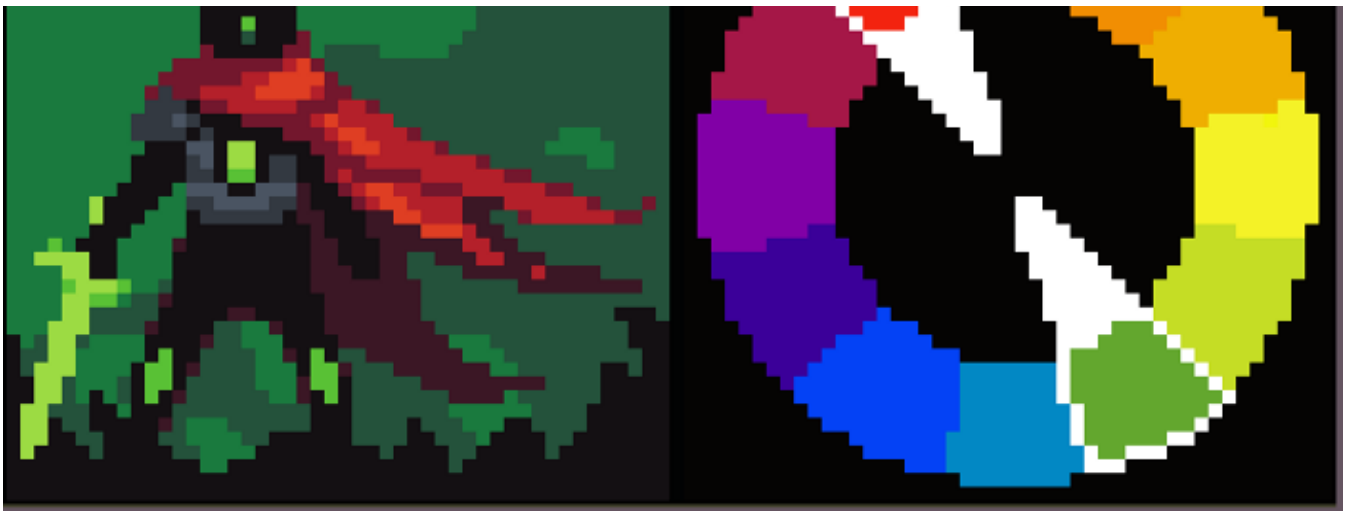
Monochromatic



Monochromatic color scheme

The simplest color scheme, you can't go wrong when you only use a single hue. This usually have a very strong and stylized feel to it, so use with caution. Keep in mind that you don't need to use a single saturation or that you are prohibited to move the hue a little, just avoid major color changes, you can still hue-shift your light.

Complementary Colors

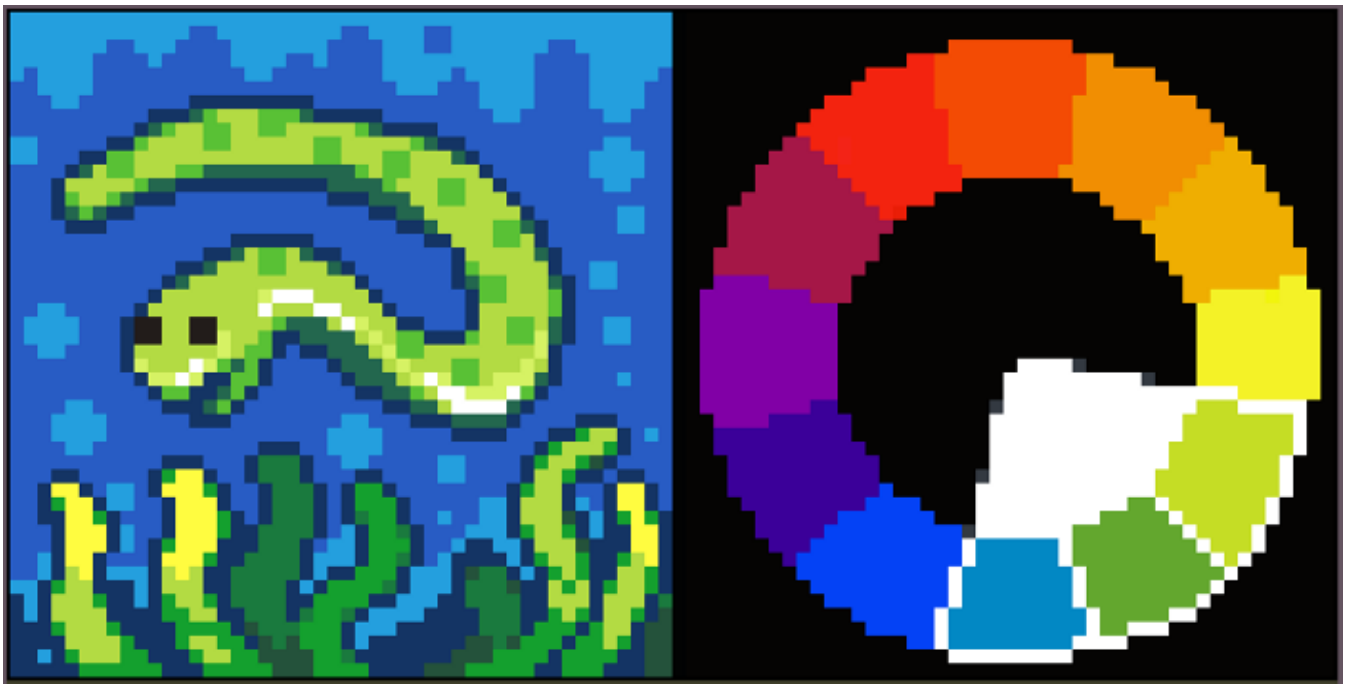


A red and green complementary scheme

This is an amazing scheme, great for creating a strong contrast. Usually one of the colors is the main one, while the complementary color, the one on the other side of the wheel, is used in some details. Red and green, blue and orange, and purple and lime are some other examples.

Careful with this combination, though. It creates some very intense and aggressive images, especially at high saturation levels.

Analogous Colors



Green-analogous color scheme

Analogous is a scheme made by choosing a main color, green in this example, and two others from nearby hues. This color scheme is usually very calming and comfortable.

It's great for low-contrast and harmonious images.

Now what?

There are a lot of color schemes, like the Triad, Split Complementary, Square, and many other variants. Color theory is a very deep subject, and if you want to learn more you can check other sources:

- On Wikipedia
- Basic Color Theory
- Thread about colors in Pixel Joint
- Understanding Colors in Blender Guru

For making your own palettes, Adobe has this really good tool: https://color.adobe.com/create/color-wheel/

And as a final disclaimer: remember, working with colors is a highly subjective thing. These are not immutable rules, just flexible guides, but I do recommend sticking with them until you feel more confident about what rules to break and how to do that.

Keep reading the part 7 here!

Design Colors Pixel Art Game Development Game Art

How to start making pixel art #7

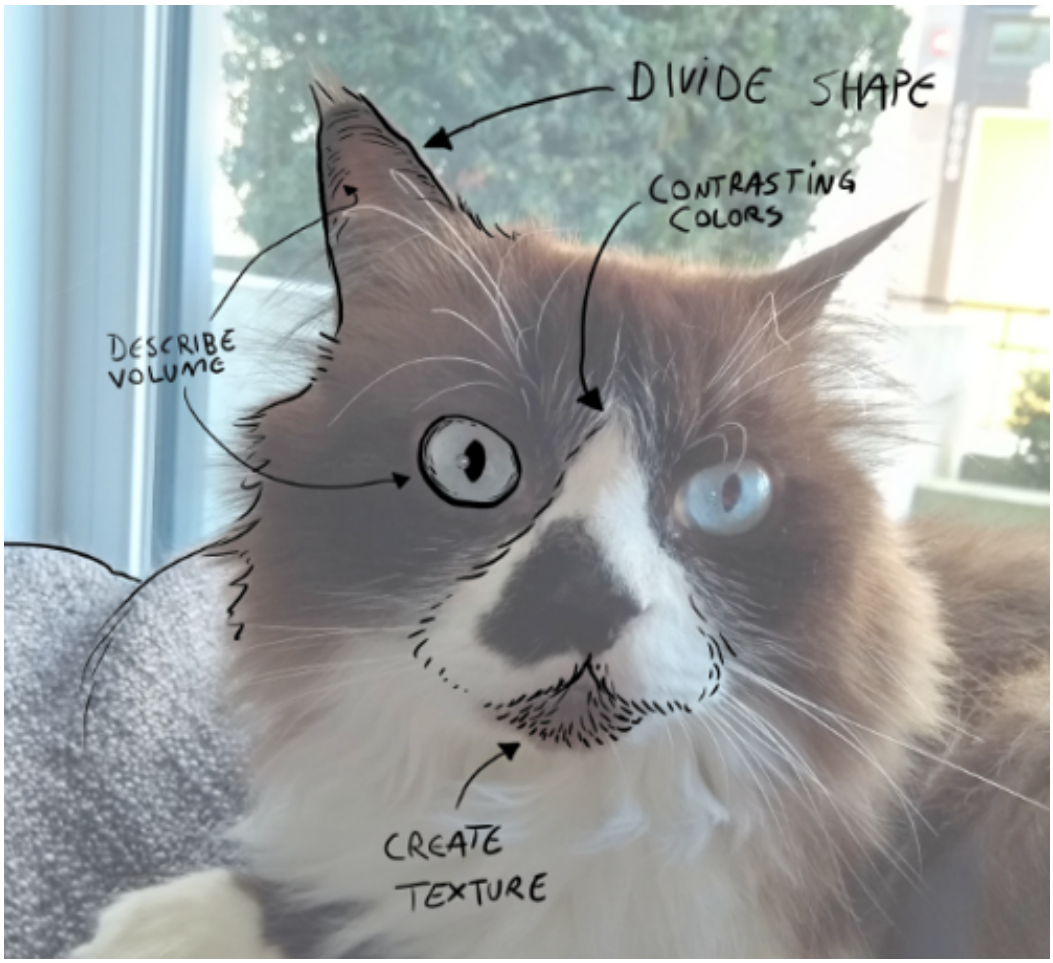
Working with lines

Pedro Medeiros Follow
May 7, 2019 · 7 min read

This article was supported by Patreon! If you like what I'm doing here, please consider supporting me there :)

Also, this is the part 7 of a series of articles, read the whole series here in the Pixel Grimoire.

What are lines



Phil explaining what lines are used for

Lines are one of the most important art tools ever invented, they are there to emphasize forms and contrast. You can't see lines in the real world, but it's something that helps us divide spaces, create texture and describe volume and direction.

In pixel art we have a very limited space in our screen. Since each pixel counts, lines can become very expensive. For this reason, if you're working with a very low resolution it's not very recommended to use lines in your pixel art. Even if you have some space it's always a good idea to really consider each line and try to get away without using them. Minimizing the amount of lines is a good idea if you want to keep your pixel art clean.

Removing unnecessary lines

You can replace unnecessary lines with shadowed or high contrasting areas. Merging the lines with dark shadows also makes the drawing look more unified and natural.

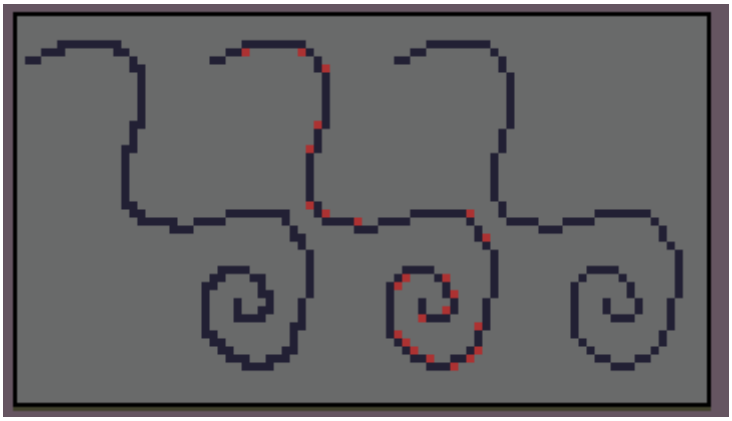
Unintentional corners

I explained this briefly before, but let's examine this problem better.

It's common that unintentional corners appear on your lines when drawing with natural gestures. This can be solved by manually cleaning them up or by using an automatic "pixel perfect" property in your pencil tool.

Pencil without and with the “Pixel-perfect” property

This algorithm is very useful for drawing single pixel lines but it’s not always perfect, you should also know how to fix unintentional corners manually.



Manually finding and removing corners from a line

It’s a simple pattern search: every time you see a corner (3 or 4 pixels forming a square shape) ask yourself if that’s really supposed to be a corner. If that part of the drawing is not a corner, remove one pixel to make it rounder.

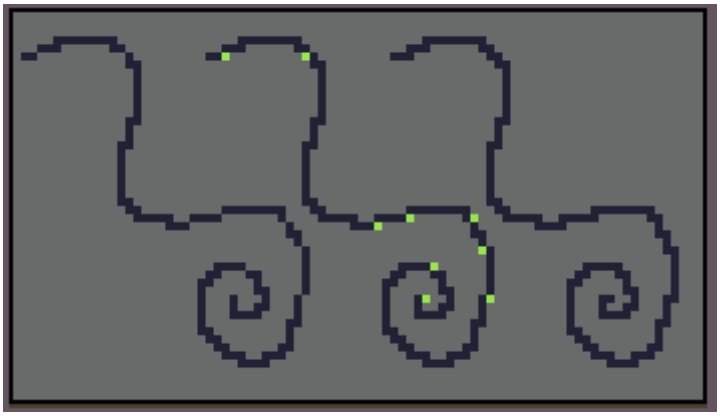
Remember that this is different from **jaggies**, and I explain what they are and how to remove them in another chapter. Fixing corners can create a lot of jaggies, so it’s usually a multiple step process: draw> fix corners> fix jaggies.

Square lines

Square lines on a stylized character

There is a different stylistic approach to the single pixel line problem. Instead of trying to remove them you can just add corners everywhere! It may sound counter-intuitive, but this is a stylistic choice that's also really interesting and works specially well with highly stylized drawings, since it simulates a thicker line.

It's basically the same process, but instead of removing the corners you should search for any diagonal pixel connection and make it square.

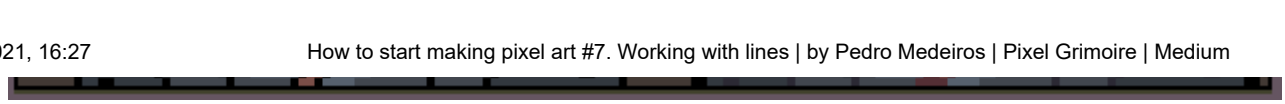


Manually adding corners to diagonal connections

Note that doing that also usually creates a lot of jagged lines, so watch out for those and don't forget to fix them.

Colored lines

Sometimes, especially in lower resolutions, lines can draw too much attention to them. One great way to soften that effect is to add darker versions of neighbor colors to them.



Black outline comparison with colored outline + double exterior outline

The process is simple, just re-visit each line you made and try to change it to a darker version of a neighbor color. Remember that when you do that, the chosen color will make that region look slightly bigger, so you might need to adjust the line position to accommodate there. Sometimes you won't be sure on which neighbor color to choose, sometimes it will be pretty obvious. When not sure, I usually go with the darker color.

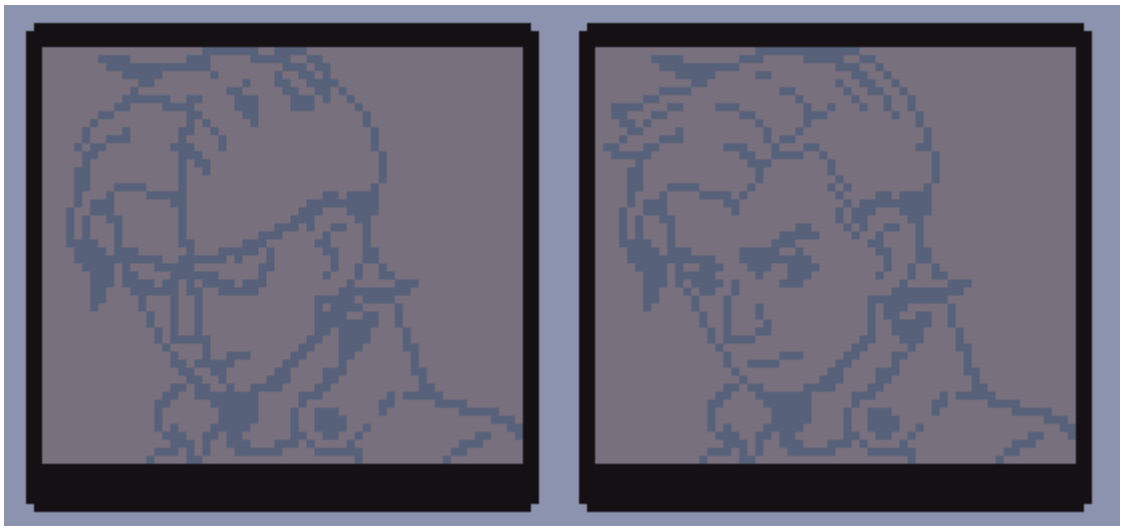
I still like to keep the exterior outline black, and depending on the resolution I even double that exterior outline. But remember, that's a personal choice, it's totally up to you.

Line sketching

We learned how to do *cluster sketching* on a previous article, now let's try something different. This approach is great for drawing characters and sketching animation frames, since it's so fast and focuses on shape and form, instead of light and color (like cluster sketching).

Let's make a case study of an actual illustration now and apply everything I explained in this chapter. If you want to follow along and practice this step-by-step, I suggest you start a new file with a 64x64 resolution and the AAP-64 palette.

Step 1 — Sketch



Step 1

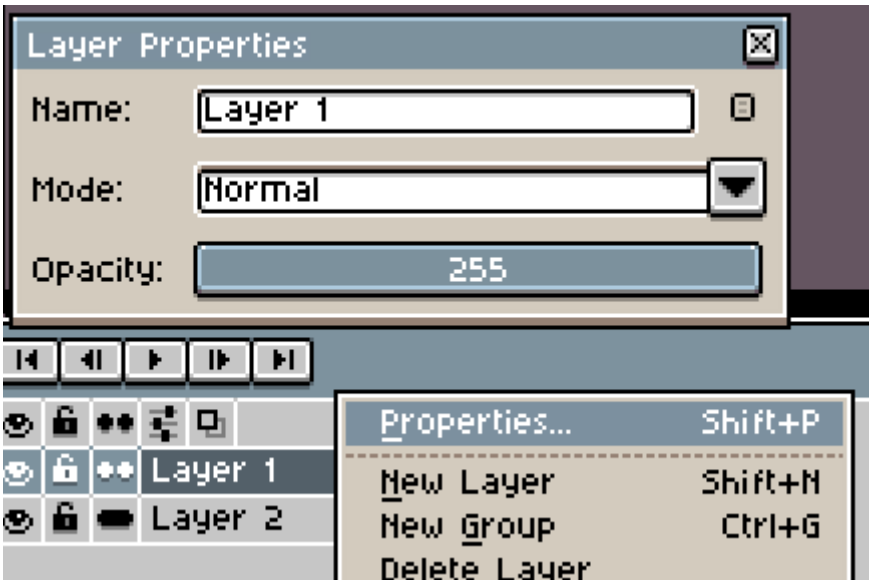
The first step is to forget pixel art rules for a bit and try to sketch the shapes as fluidly as possible. Basically the same thing as drawing on paper or in other digital, non-pixel

art medium. Just try not to worry too much about cleaning up the lines yet. After that you can clean it up a little, removing obvious guide lines and defining some shapes.

Step 2 — Cleanup

Step 2

Now let's cleanup. Make the sketch layer semi-transparent by right clicking on its name and choosing properties, then lower the opacity. Now you can just draw on a layer on top of everything.



Lowering the opacity

The idea now is to make the final line art. Focus on drawing the important lines first, the ones that make your character's silhouette and face, for example. Don't worry about color or light here, you can draw some of the main shadows if you want, but your focus is to get the major shapes right.

Step 3 — Base colors

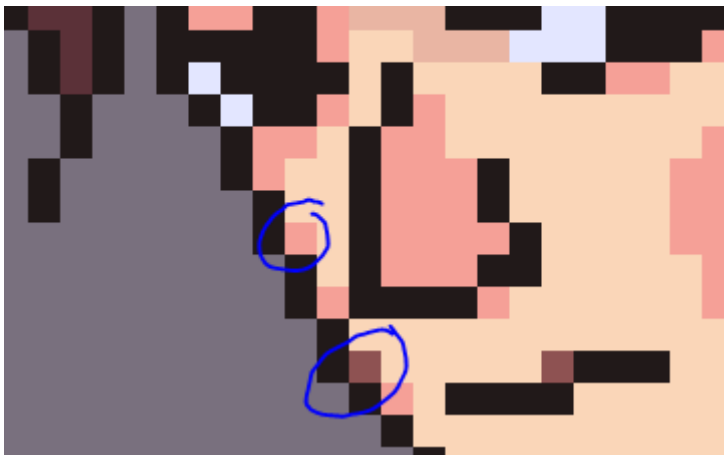
Step 3

This is usually a simple step, just find colors that match with what you want and fill drawing areas with the paint bucket tool. Try to avoid colors that are too saturated or too bright. It's not the time to paint light and shadow, but be mindful of how the image is supposed to be lit, sometimes areas can be affected by their surrounding colors.

Step 4 — Shading

Step 4

Start shading the image, adding both brighter and darker areas. Try to imagine how the light would shine on the object you are shading and render it accordingly. Having reference images and photos for this step is also a great practice.



Softening the lines with small shadows

I like to try to “soften” the black outlines as much as I can during this step by adding darker colors on the curves, almost like an anti-alias. Don’t overdo this, though, or the lines can start to look blurry.

You can change the line art on this step, but focus on the shading for now.

Step 5 — Coloring lines and anti-alias

Step 5

Like I explained before, too many black outlines can get a little distracting. Softening them by making them colored is usually a good idea. Just search for the nearest and darkest color nearby and change the outline color to a darker version of it.

If the outline separates very distinct shapes like the chin and the neck, or the colar and the hair, you can keep the line very dark or even black. That’s the reason why I like to keep the external outline of the character black. Soft transitions usually have lighter lines, like the nose lines.

Step 6 — Polish and detail

Final image

The last step is where you should make sure that all the lines are anti-aliased, all the shadow transitions are properly softened (or not! If they are hard transitions) and add all the small details.

Add backlight, small specular lights and reflections in this step. Since everything is already in place, you should be able to measure how much detail you want in your image. Try not to overdo this, and watch out for too much noise.

And we are done! Keep practicing and try to mix this with concepts you learned in the other chapters.

Good luck!

Keep reading the part 8 here!

Design Pixel Art Tutorial

How to start making pixel art #8

Saving and Exporting Pixel Art

Pedro Medeiros Follow
Jun 6, 2019 · 9 min read

This article was supported by Patreon! If you like what I'm doing, please consider supporting me there.

Also, this is the part 8 of a series of articles, read the whole series in the Pixel Grimoire. I'll get a bit technical here so feel free to skip paragraphs of things you are not interested in.

Bitmap and vector

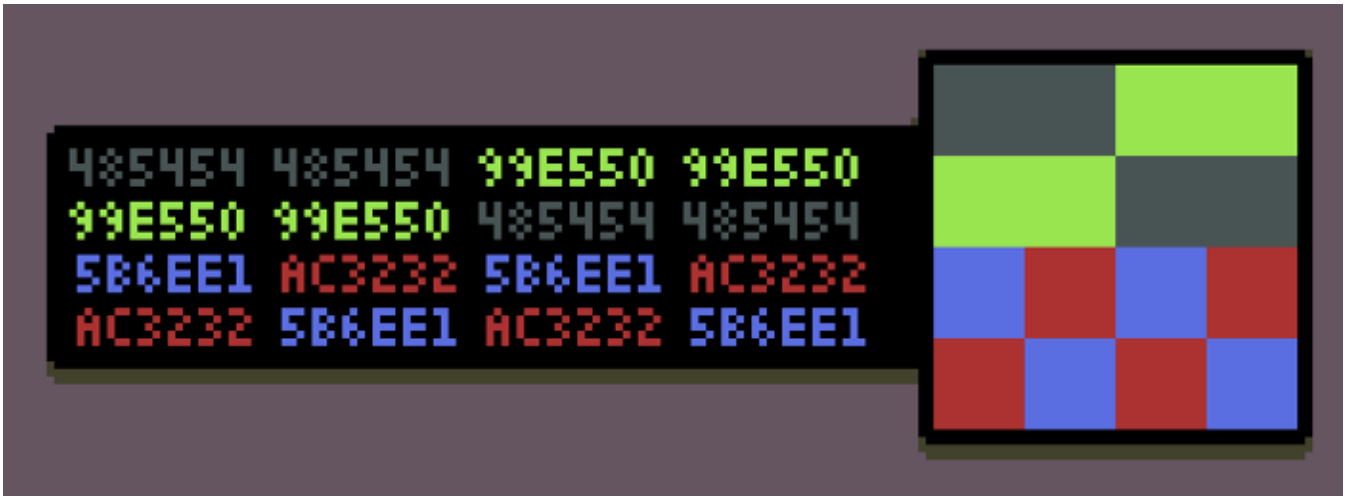
Let's start talking about how images are saved in the computer. If we simplify a lot, there are basically two types of 2D image files: bitmap and vector.

Vector images are mostly a collection of points, line coordinates, and color information, which makes them very useful por creating images that can be rescaled into pretty much any resolution. The main drawback is that it's hard to use it for detailed images like photos, and you have very little fine control on the pixel scale. While it's possible to create something that looks like pixel art on vector, or exporting pixel art into vector format, it's rarely done. Some edge cases exist, like printing pixel art to a really big billboard.

```
<circle cx="50" cy="50" r="40" stroke="black" stroke-width="3"
fill="red" />
```

Typical text inside a SVG file and the resulting image

Bitmap images, on the other hand, are the reason why pixel art exists. Bitmap imagines the images as really big grids that are printed to the screen, and saves each pixel individually.



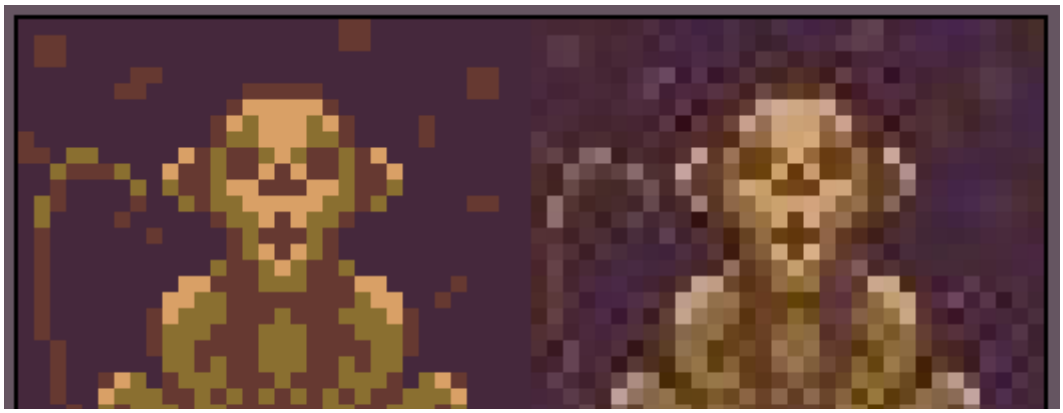
A simplified example of how a bitmap is saved

Saving a file this way makes it very big sometimes. A 1080 x 1920 image means that 2.073.600 individual pixels need to be stored, so sometimes it's clever to compress that image using cool algorithms.

Image compression

Image compression was created to reduce image file size, which is great, but depending on the way you do it it's possible to damage the file.

Compressing can be lossy or lossless. Lossy compression is usually fine if you're working with photos or even large illustrations, but not cool if you're saving pixel art. Saving a pixel art image as JPG will doom it to random color changes and weird image artifacts. When compressing our pixel art we should always check if we're using a lossless compression, such as PNG.



PNG (lossless) versus JPEG (lossy)

As a general rule, the image format supported by the editor you're using, like Aseprite's **.aseprite** or GraphicsGale's **.gal**, always keeps your file safe and with all its layers and metadata.

Saving in Aseprite

In Aseprite you can save your file using the save dialog (Ctrl+S, or Ctrl+Shit+S for saving to a new file).

Aseprite's save dialog

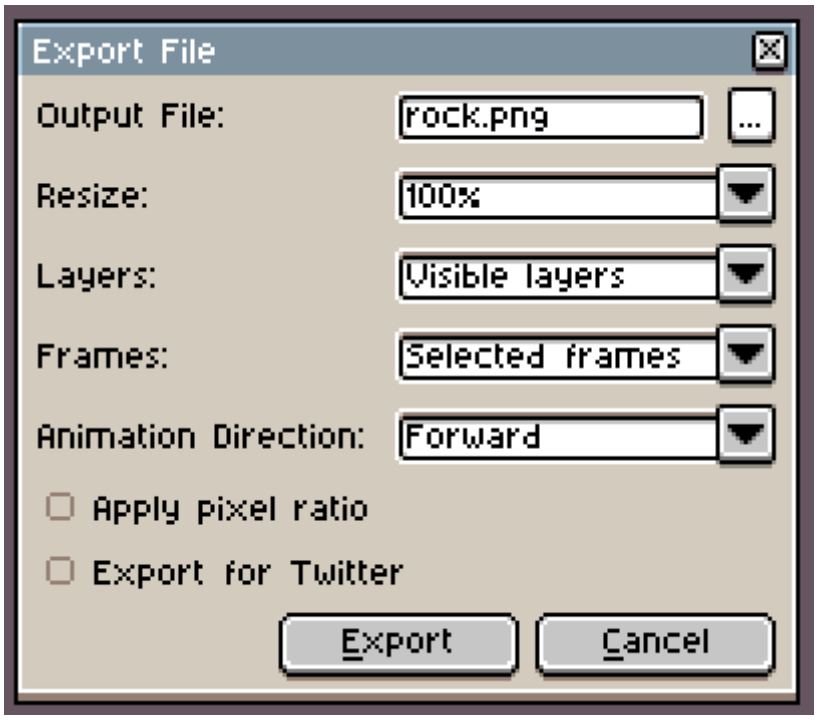
Pretty intuitive, right? Use this to save your files in **.aseprite** format and keep all the file information, so you can can come back to it later if you want to make changes.

Exporting for preview

After you finished your sprite you might want to post it somewhere or send it to someone, but if you just save it as **.png** two bad things will happen: one, your image is saved in a flattened file, which means no layers or meta-data, and two, your png probably looks something like this:

This rock is too small to see any detail

The correct way of saving an image for preview is to use the **export** dialog, which can be accessed in File>Export menu.



Aseprite's export file menu

This dialog is great and has tons of useful options. Let's go one by one:

Output File: The filename. You can simply type the file extension you want to use and Aseprite will save it in that format. I recommend using ".png" for static images or ".gif" for animations.

This will not change your *Save* command to use this file as default, so don't worry about choosing a format that will flatten everything.

Resize: Pixel art is usually too small for today's monitors, so we need to resize it. Aseprite makes it easy by giving you multiple resolutions.

Resizing in whole numbers versus resizing in fractions

You probably noticed that you can only increase the scale of the image by 100%'s but no fractions, like 250%. This is because if you scale it to a number that's not whole, your pixel art will look broken and weird. That's a very strong limitation of the pixel art medium and there aren't many ways around it. Always resize your art using whole numbers (200%, 300% and so on) and never fractions (130%, 250%).

If you really need it to fit somewhere, try resizing using the closest smallest whole number and then extend the canvas to the size you need.

Usually 200% or 300% are good enough for posting online.

Layers: When you export a file you will flatten it, which means it'll lose all layer information. This is a handy property that allows you to choose which layers you want to keep.

Frames: I'll get into more detail in the next section, but here is where you choose which frames you want to export. If you have more than one frame, two things can happen:

- If you save in a format that supports animation, such as **.gif**, the animation will be exported to this file.
- If you save in a format that doesn't support animation, such as **.png**, Aseprite will save it in multiple files, one for each frame.

Animation direction: Here you can see the most common directions like "forward", "backward" and so on, and pick the one that's best for your animation. They should be pretty intuitive, but if you want to know more, they are very easy to experiment with.

Apply pixel ratio: This is not very common, but if you're using a non-square pixel, this will flatten that to square pixels. If you're not sure what that means, don't worry, it won't change anything for you.

Export for Twitter: A handy checkbox that makes the last frame of an animation have half of the duration, which makes the twitter mp4 loop perfectly!

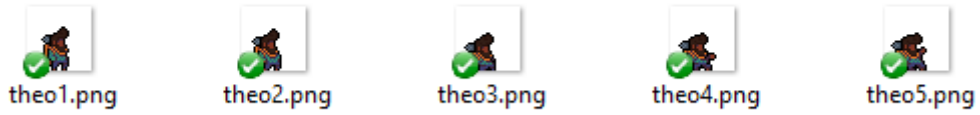
Saving for the engine

Game engines are vastly different and each one will require you to do specific things, but there are usually some common practices:

- Let's start with the file format: while some engines will accept **tiff**, **gif**, or even **bmp**, it's usually a good idea to use **png**. It's light, relatively fast to unpack, and has a really good transparency support. Just don't use **jpg**, **mp4**, or any other lossy compressed format.
- Unless you have a really good reason, **never** ever scale your pixel art for exporting to the game engine. Ideally the sprite should be scaled by the engine and all files should have their real resolution.
- Animations should be saved as image sequences or sprite sheets. Do not attempt to use any video format, unless you are making something very specific.

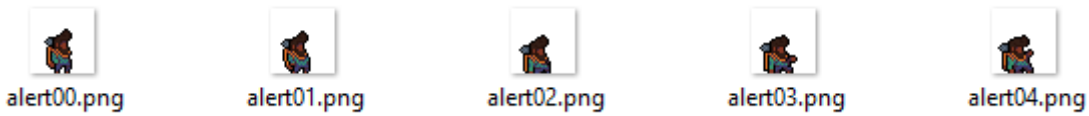
Saving an image sequence

When saving an animation as **png**, Aseprite will create an image sequence, which looks something like this:



Simple animation exported to png on Aseprite

Sometimes that's enough, but I prefer files that start on zero, and I also like having two digits. Fortunately that's also very easy to do: in the filename field, just type the name of the file followed by "00". In this example, I named my file **alert00.png**, and this was the result:

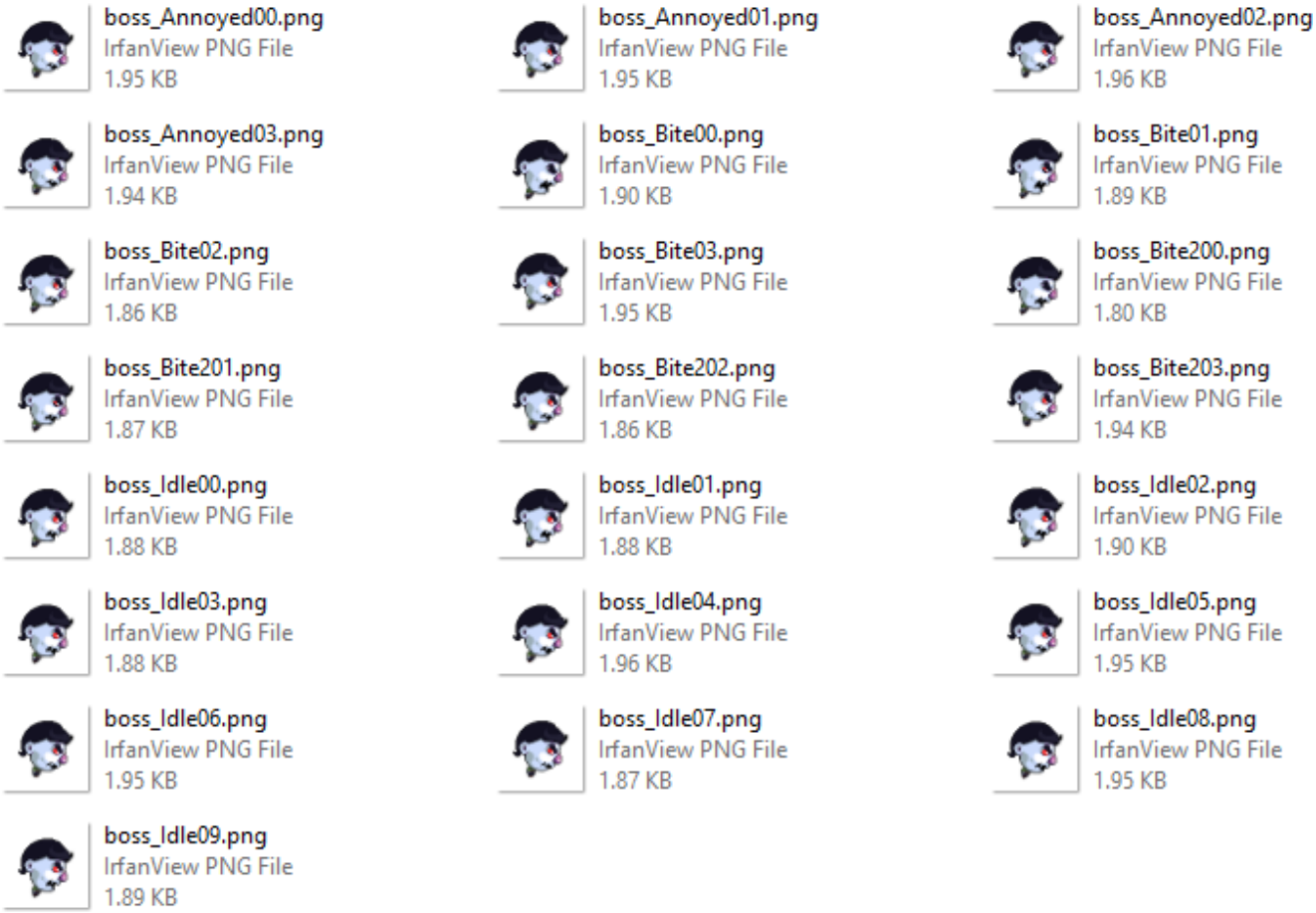


Feel free to experiment by adding different numbers in the end of the file.

Another **very** useful trick is working with multiple animation tags. In Aseprite, you can select groups of frames and create tags. This way you can have a single file with multiple animations. It looks like this:

A timeline with multiple animation tags

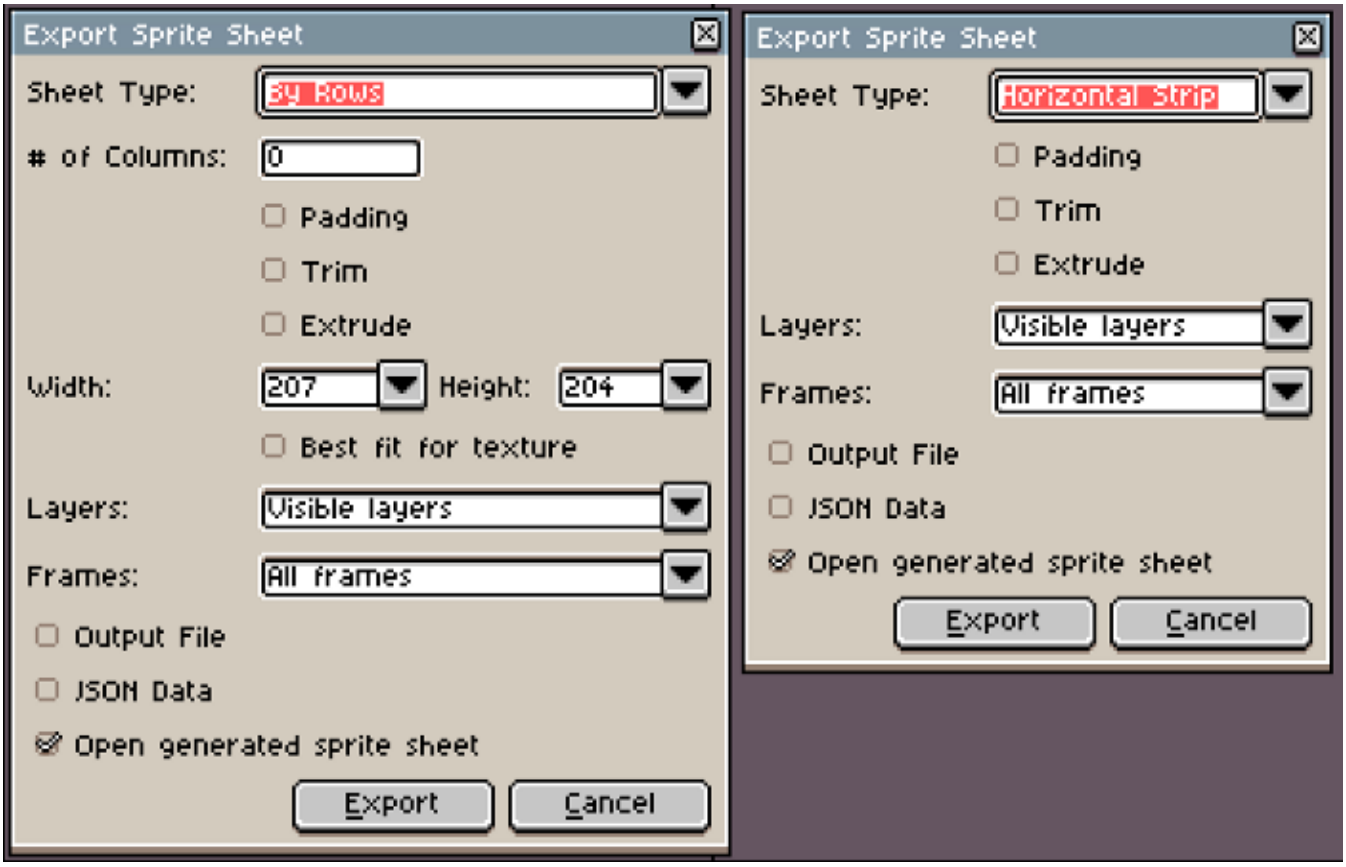
That's great for organizing, and exporting it is also very easy, just type something like **boss_{tag}{tagframe00}.png** and Aseprite will export files like this:



Useful, right? If you want more or less zeros, just change **{tagframe00}** to **{tagframe000}** or even just **{tagframe}**.

Saving a sprite sheet

Some game engines prefer sprite sheets instead of multiple images, and we can export them automatically using the File>Export Sprite Sheet command.



The Export Sprite Sheet dialog with horizontal strip and grid by rows selected

Here we can select various options and most of them will be dictated by the engine you're using. Some require padding between the sprites, JSON descriptors, others support trimming the transparency. Extruding makes that all tiles have their border colors extruded around one extra pixel, to avoid seams on 3D tiles, if you're working on a 3D engine. When not sure, just don't check anything.

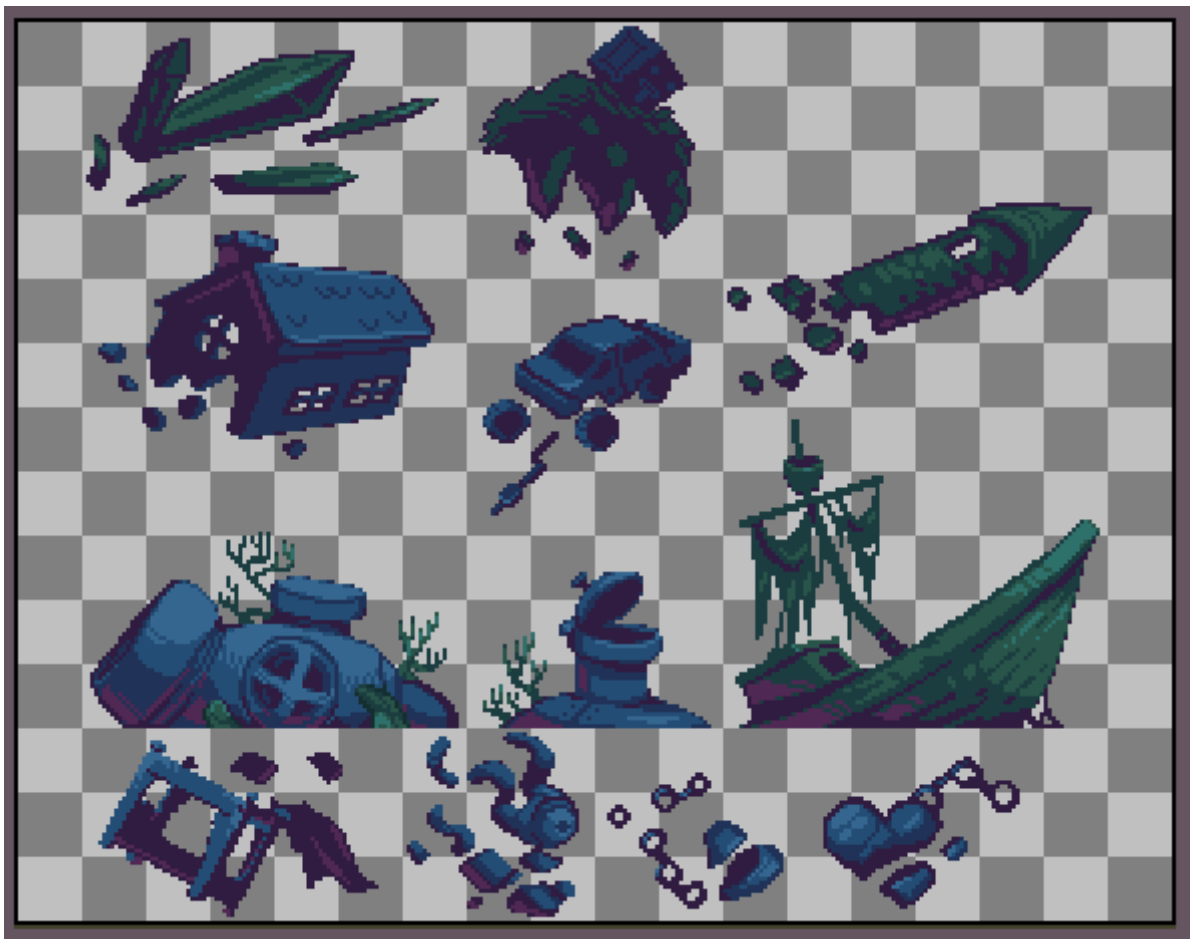
About the number of rows and columns, if your engine doesn't require anything specific I recommend checking "Best fit for texture" and let Aseprite calculate the optimal size.

One thing to notice about this screen is that Aseprite will **not** save the file if you hit export unless you check the "Output file" checkbox and choose a filename. One alternative is to check "Open generated sprite sheet" which will open it in a new file that you can manually save later, if you want.

Slices and the command line

One last very useful trick is to use slices. Slices are great for making multiple images in a single canvas, I really like using them when making props for a game scenery, so I can always have the other props nearby for comparing size.

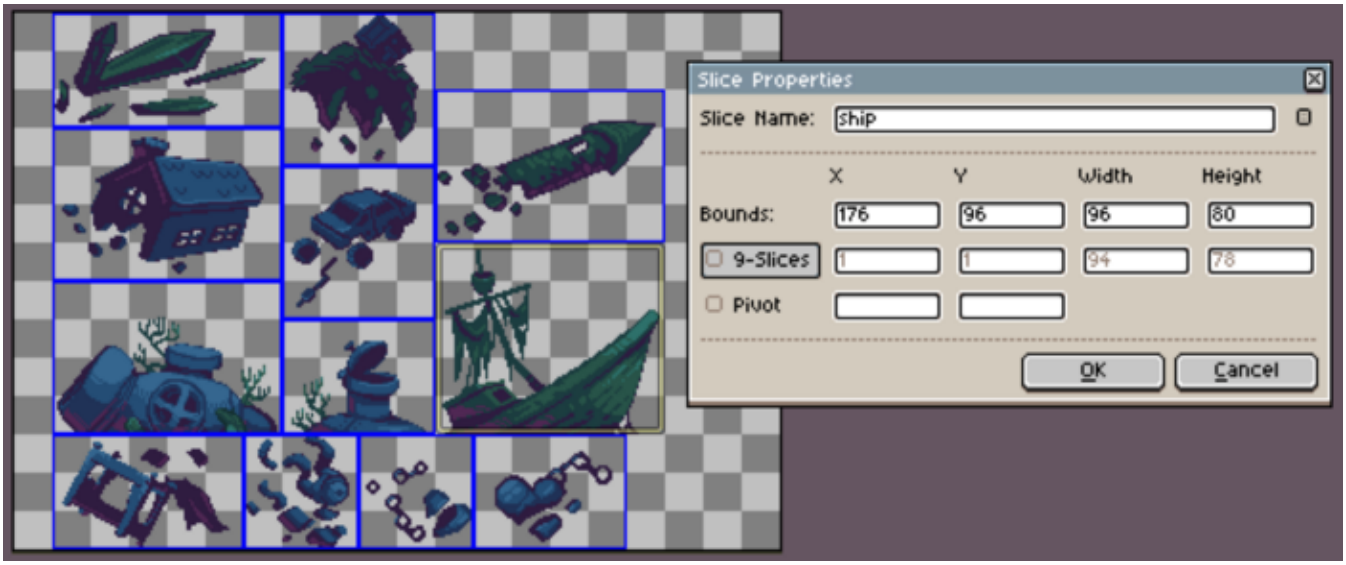
The first step is to make the sprites in a single file, side by side. Just don't let them overlap each other in a rectangular area.



Props side by side drawn in a single file

Next step, select the slice tool and start making selections around each object, be careful to not include any other sprite parts in each selection. After that, you can

double click on one (or right click and see the properties) and choose a name for it.



Slices done, one selected

Now, the command line! If you're mainly an artist, this part can sound a little scary but I promise that after you get the hang of it and set everything up, this will automate a lot of the boring work when exporting files.

First you will need to have Aseprite in your PATH, if you are unsure what I mean by that, follow this tutorial and add the Aseprite.exe path to your environment variable "PATH".

Now, save your Aseprite file and open your favorite command line tool on that file's folder. If you don't know how to do that, on Windows you can Shift+Right click on a folder and select "Open Windows PowerShell window here".

And then we do the fun part, the actual command line! Just type this (replace *myFile* with your filename) and press enter:

```
Aseprite.exe -b '.\myFile.aseprite' --save-as '{slice}.png'
```

Let's break it down on what that actually means:

- `Aseprite.exe` is the command to start Aseprite.
- `-b` means we are in batch mode. If you don't add this, Aseprite will actually open, and we don't need that.

- `'.\myFile.aseprite'` is the filename and extension of the file we're working with. This path is relative to the current path.
- `--save-as` is the command to save the selected file. This command also formats the file name with the tags you provide.
- `'{slice}.png'` is the filename, *{slice}* is the tag that will replace the filename with the slice name and crop it.

If you want more information on the command line, Aseprite has a really good guide on how to use it, check it here!

Now what

Now you know some of the practices on how to properly save pixel art! When not sure of something, use this as reference:

- Always rescale in whole numbers, never in fractions, never reduce.
- Aseprite will save image sequences automatically.
- Stay away from **jpg**, **mp4**, and other formats that use lossy image compression.
- When not sure, just go with **png** and no scaling.

Still working on part 9! Check my Patreon for now :)

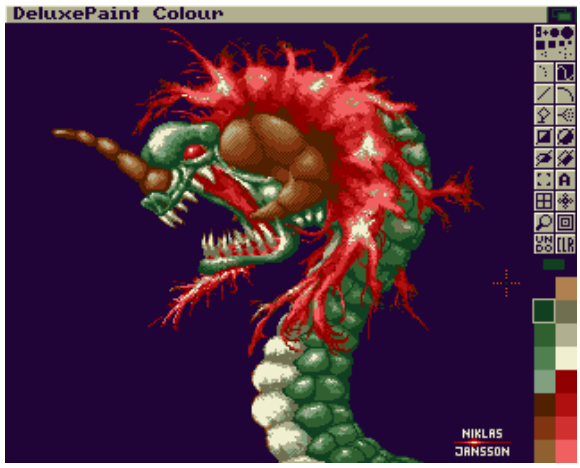
Design Pixel Art Tutorial

This page is maybe 26.2% done! Lots of placeholder blurbs!

Pixel Art Tutorial

Preramble

I don't think I quite understood the nature of pixels until I got ahold of Deluxe Paint in the late 80's. I knew about pixels since the C64 days, but never access to any kind of editor which allowed me to see what was going on behind the distortions of the TV shadow mask, scanlines, phosphor glow and RF-modulator signal compression (distortion).



"GHERKING" was found lurking on one of my old Amiga floppy disks. It was done in Deluxe Paint III sometime in the late 80's or early 90's (note dragon-mullet). It's less an attempt at pixel art and more an attempt at "awesome picture" (demo scene style). Nowadays I would not attempt to pixel something this big - I'd just paint it instead.

The first thing I pixeled was probably a hammer from the Super Mario Bros manual (which for some reason had line art pixel drawings in it). Before this, I had actually drawn tiles from games like Megaman on cardboard and built my own levels, but working with pixels inside of a computer gave me a much stronger feeling of being privy with the game world realm.

I spent a lot of time working on projects in Deluxe Paint, mostly experimenting with my own techniques because no one else around me was doing pixel art. Trying to see what games were doing with the pixels and frames while playing a game was hard. Ripping reference graphics from games was a hassle. I remember trying to use "3rd Day", a program which would analyze the residual RAM contents after a reboot, but it was difficulty to get results. I also did a bit of programming and released a handful of tiny games and silly things into the local geek community (I didn't really know what a BBS was back then).

I work in Photoshop nowadays, even though it's in some ways inferior to Deluxe Paint. There are other modern programs out there for doing pixel art, animation and tiles specifically, but I haven't really bothered with them. I've adapted my work methods so I can take advantage of the features which PS does offer.

Approaching pixel art as a painter

Painting for painting's sake is not something I have done much of recently (is that where people encounter the infamous "artist block"?), but I noticed early on that I was much better at painting than at drawing. I don't think I started to address that imbalance until mid 200X, and I think my pixel art benefitted much from it.

Computer graphics is something I have enjoyed doing since before I got my hands on a computer. Back then, even professional developers (such as Miyamoto) drew the graphics on grid paper first. In a way, pixel art is like analog art scaled down, but the pixel space can force certain exaggerations, detail reductions, alignments, graphical simplifications and iconographies. It has to be adapted so it's readable, but a lot of the functionally effective ideas and wisdoms from analog art (be it more abstract graphical ideas or realistic-ish painting styles) can be ported and applied, I think.

However, when a human is presented with the very structural pixel art space she tends to ignore that and go into telephone doodle mode, making patterns, symmetries, outlines around outlines, little

pointless ramps of colors, and other needless additions because the spatial forms somehow encourage it. When I fall into that mindset it helps to retract my mind from the pixel space and imagine the sprite/tile more like a regular painting done with a very large brush. Each pixel is very important as it actually corresponds to a large blotch of paint rather than a single tiny pixel. I think pixel art is very much about knowing where to make sacrifices of detail due to the scale, and how to make each pixel do double duty.

Anyways, I did write that tutorial which mostly concerns rendering/painting, so naturally people have asked me to write one about anatomy, and mechas, and pixel art. With this tutorial I won't be covering painting in general, but rather be focusing on the quirks and qualities which are unique to pixel art. I'll also be talking about some of the early hardware limits which still can be used to impose useful (if not enjoyable) restrictions on the artist.

Index

>>> I'll put the index list with anchors here if the page grows too long. Will be useful for people who want to link a certain topic.

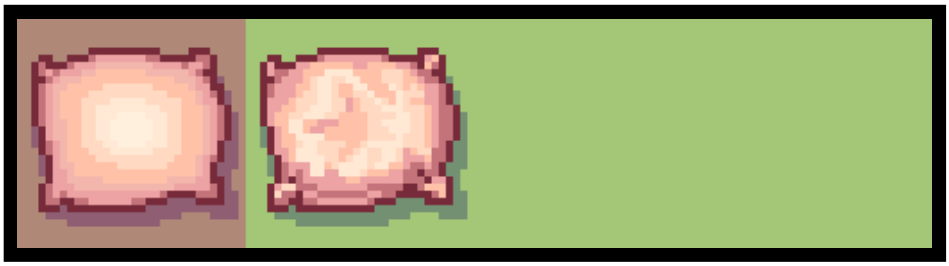
Working with pixel art in Photoshop

>>> This (more technical) section should cover:

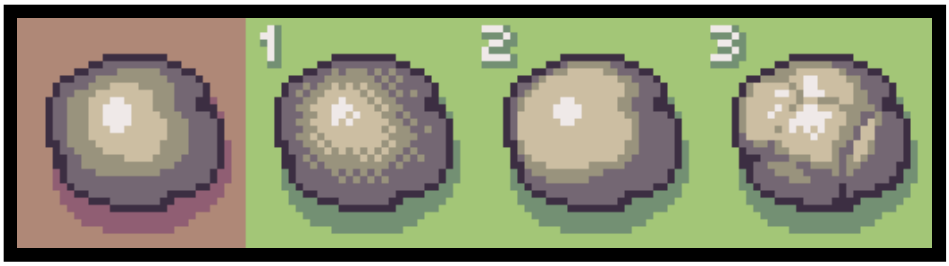
- Using a zoomed in work view and a 2x "take a step back and look at it" view.
- How to test tiling and fixing tiling seams. Setting up a grid. Using PS/Filters/Other/Offset or copy pasting manually.
- Painting pixel art using opacity, still using a palette (external in another window).
- Working in indexed mode.
- Short on the usefulness of the .raw file format for doing map editing.

Some practical examples

>>> This section will take a look at what mistakes people just getting into pixel art tends to make the most. This way people can directly engage a topic, rather than reading general advice and try figure out where those can be applied practically. I'll cover:



We start out with some pillow shading compulsions and inappropriate sculpting of three-dimensional form. Pillow shading can look a bit like a pillow, with a soft radial gradient ramping up towards the center of the form. It can make the form look uninteresting or irrelevant to shape of the object. The solution is to make the gradient sculpt something interesting. This requires some painting skills of course. Put briefly, building up value (lightness) then terminating it abruptly against the shade of another gradient can be an effective way to break the pillow shaded look.



When working with a limited palette you can get banding, i.e. visible bands of color. You might think using more colors to make the gradients look smoother is a good idea, but this can be a trap because pixel art actually often looks better with a restrained palette. So how can we remove these bands?

Banding draws attention to the borders between color fields (rather than the center of the bands themselves). It will make the eye think, hey, this area is important, there's contrast here. If the area isn't actually important then it's a distraction that should be done away with.

Solution 1: We just dither, but dithering is a solution that I prefer to use sparingly, and I find it's more effective when I can make it do double duty - create a gradient AND portray an appropriate texture. Dithering can look a bit wonky if applied to things which we expect to be smooth, but I suppose it can be used in an artsy, brushstrokey kind of way.

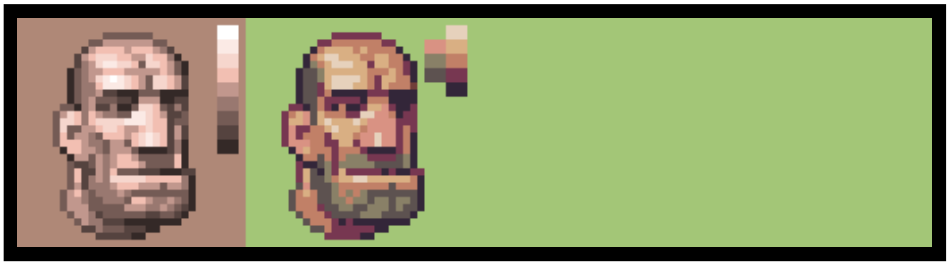
Solution 2: By compressing the bands the value change is over so quick the eye doesn't quite see the borders. Also, the banded edge is best placed near a natural looking "terminator" (where shadow begins) because then the eye expects it and it becomes invisible and not distracting. I think screen print faces some of the same banding problems which pixel art does. If you're gonna have a contrasting edge between two color fields, might as well try to place it where it makes compositional sense, and/or describes a shadow-light plane change, or perhaps just a seemingly natural change in value/coloration.

Solution 3: If your shape has potential for detail you can break up the gradient in little manageable clumps.

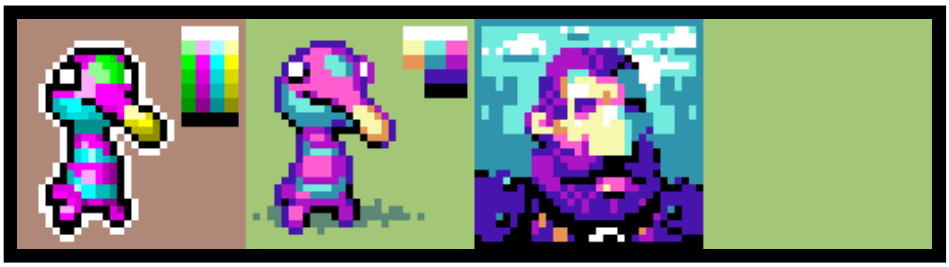
Another illustration.

Strange light sources such as flat looking rim light / rim shadow can be seen as a variant of pillow shading. I prefer to use a front-side-top light myself as it describes form pretty nicely, but there are situations where you might want to use fancy colored back/rim light to set mood (such as in a portrait).

Excessive Anti-Aliasing can make a piece look blurry and indistinct. Pixel art is more about being graphical than being able to, say, make a circle look stunningly smooth. You have to make every pixel count and when your AA starts to creep out you lose the... pixel space efficiency and graphical clarity which I think is at the heart of pixel art.

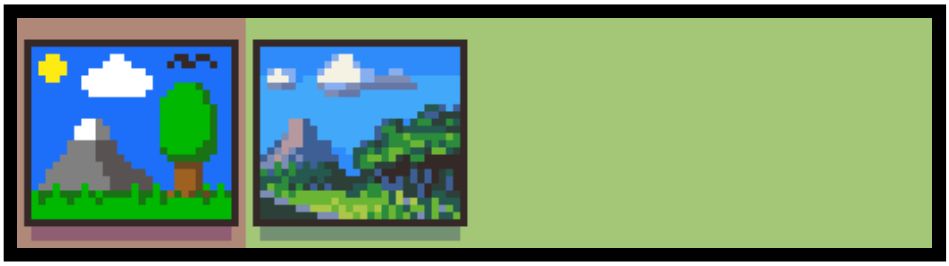


Unless you are working with very small palettes (say, 3 colors), you probably need to set up color ramps (gradients) at some point. When creating your ramps I'd advice against mixing in portions of black and white to make the ramp's ends. In nature, the hue of a color often changes as it goes from light to dark, so a black-pink-white ramp can look muddy and artificial. On top of this, an object, such as a human face, often has different hues on different parts. For human skin tones, I like to make the shadow a grayish warm purple, keep the mid tone orange, and the light almost yellowish. When doing outdoor scenes I also mix in some colder grayish tones into the shadow (because of sky ambience) and yellow into the light colors (because of the warm sunlight).



There are many colors, but I've found that some are more usable than others, especially when doing realistic looking stuff. I'd avoid building long ramps from super (fully) saturated colors, unless you're doing a neon piece or some kind of more graphical excursion. This doesn't mean that your colors should all be dull or pastel - highly saturated colors can be very effective at adding extra dimensionality to certain spots, but I think their frequency in the palette should be proportional to the likelihood that you're gonna use them. Here I drew a strange little neon creature because it's within the scope of the palette. When doing stuff like faces or nature I'll have to take a more palette-artsy approach.

Unrelated, outlines around outlines. I don't like them much, but I suppose they have their use when conveying effects like... shields and auras. It's tempting to throw on an extra outline because it looks kind of neat and splashy, but it's very much a temporary love affair, I've found.



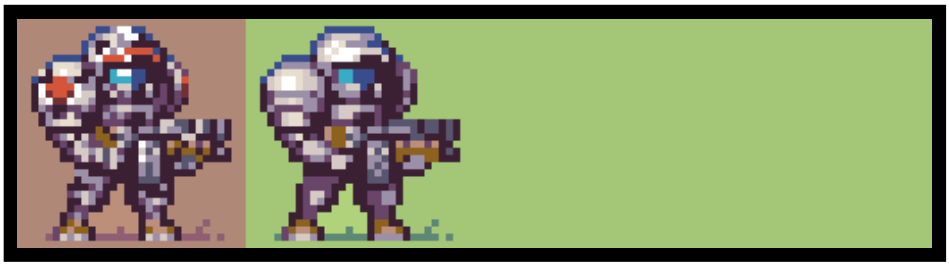
The use of plain local colors (the color an object is locally, in neutral light, or perhaps in a cartoon) can be effective when working graphically because you need to be clear about what's a tree and what's a mountain. But, when doing a more painterly piece you don't have to make every tree trunk brown and every mountain gray. Colors are shifted in all sorts of directions in nature.

Slightly related, doing template objects results in a lack of style. On the earlier piece I avoided doing regular trees with brown trunks. Here I'm trying to get away from the silvery box-robot. This is a bit of a tug-o-war when doing pixel art. One one hand, your designs need to be very clear, and on the other hand you want them to be a little unfamiliar so they are interesting and new to the viewer...

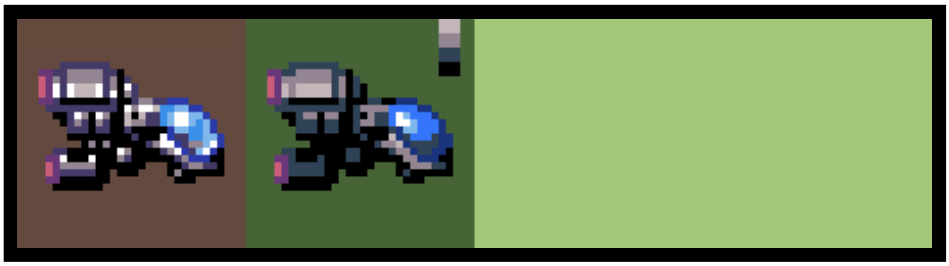
Excessive black lining might arise from some sort of compulsion to mark and separate every detail. When colors are surrounded by the black they look darker and a bit muddy. You also lose pixel space to the lines and the whole figure is flattened because the lines seemingly pushes every detail into the same plane (especially if the color ramps used are very low contrast). Another way to separate details is by using contrast and lighter additive lines (black lines are subtractive). The style that I use benefits from a mix of subtractive lines, lost lines (somehow suggested, but not drawn) and additive lines.

Another consideration when doing black lined figures is that the lines at the base of a shape looks a bit like a shadow and can make the whole figure seem like it's floating. Once removed, the figure seems to grow out of the ground. However, if you skip the line art altogether you can gain much more room for internal details, but then you have to rely on contrast against the background for the figure's silhouette to read.

Quickly drawn pixel lines will probably look jaggy and disorderly, not unlike a pencil sketch done by someone who can't draw and is rubbing the lines down too much (barbwire lines). Certain details does warrant quirks in the lines, but often you want to use the 1,1,2,3 or 2,1,2 (or some such) kind of curve. On the shadow side of object I sometimes fill in the diagonals (see boulder higher up), but it can look a bit untidy.



Noise and needlessly orphaned pixels is what you get when you try to cram too many details in. It's tempting to put as many details as you possibly can in there, but you have to consider the scale at which you're painting and how to be clear, and also ease of animation (details convenient on one frame might be hard to replicate in the next, slightly rotated frame). You might know the intention behind specific pixels, but the viewer generally doesn't so she'll more likely see them as noise (and so will you once you come back to the piece later on). That said, it's nice to have some noisy areas to provide variation in texture.



Adding little highlights here and there is super satisfying, but I think it does this already greebly spaceship a disservice as it becomes very noisy. Try to pick a few sweet spots for them and prune the rest. Here I'm trying to give examples of the two extremes. There are (or were, back in the Amiga days) chrome-happy styles which embrace highlights and make them work. Keep in mind that a flat red would become some kind of shiny pink when highlighted excessively, i.e. it will lose its color identity. Dull materials are probably inappropriate to highlight unless you're doing strong light source stuff. When it comes to gloss materials I prefer to make the highlight pop with a few pixels, rather than building up to it with a long ramp. Metals (& chrome) gets the longer ramp.

Certain things are difficult to build out of LEGO bricks, and the same goes for pixels. Daleks might be one example. They have the flat, sloped faces forming the round skirt, and a million little round bumps. If I have the choice, I just avoid doing the problematic parts of the design and change them to something which is more compatible with the pixel space.

Anyways, someone sent me these Dalek sprites and asked me if I could do something with them, so I quickly applied the style I've been using a lot on this page so far. It can sort of be summarized by the gradient shown above the Not-Daleks. Note the compression of the terminator/edge and sparingly used highlights. The design was moved away from Dalek to something less detailed and more chunky. I kept the conical shape but avoided the longer slopes by breaking them into steps of 90 and 45 degrees.

This one I call... What are you trying to do and who are you trying to impress? The palette has a bunch of low contrast grays and seemingly random hue shifts in the "color" ramp, then BAM, full black. And two useless greens with no larger difference. And what's this compulsion to do little pixel patterns? Perhaps our artist here saw a transparent and oily looking eel before his inner eye, or perhaps he was just mindlessly throwing down some pixels.

Now, there's a chance the strange palette can be used for something (like a moody misty lighthouse shot), and we stumbled upon a fresh aesthetic with these subtle hues and reoccurring structures taking command over traditional forms, and the black point topping it off, creating an unusual return point for the human eye.

I thought about doing fixed version, but this piece has no set form which it strives for. The Eel stands defiant on an inherent pedestal labeled "To change me is to violate me" ?

Pixel level optimizations

>>> Write about the finer points of pixel to pixel relations.

- Micro-banding (e.g. staircases).
- About pixel suitable alignments of detail and line art. How this can affect style (Take a look at Z from Bitmap Brothers for the angularity, or Dodonpachi).
- Use of diagonal displacements to squeeze in more details per line (e.g. Gameboy RPG characters).
- Using dither for shading versus texture (e.g. chainmail). Also note dithering used for separation of ground planes in Chaos Engine. And, it can be used to achieve duality in colors.
- Using orphan pixels for texture, and...
- More about partially lost outlines.

Enter... the gem hunter

Blentard the gem hunter has found a gem, but he needs to take it home and polish it. I'll guide you on your way, Bloenard!

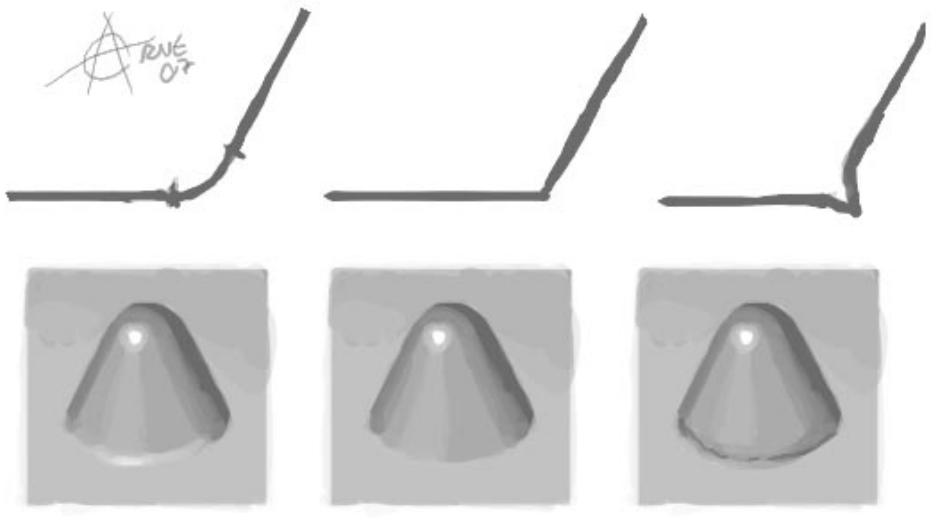
Here I've crafted a piece which exhibits many of the common mistakes listed above. I have the dirty strange gradients going towards white, and details lost in these gradients, such as the roof tiling. The mountain has a mono-gray pillow shading whilst the trees have gradients with banding (going upwards). The grass is noisy and the lines are jaggy. The sea is blue, the grass is green and mountains are gray, like in a kid's drawing. I've attempted to give the human teeth, pupils and a bunch of details which become unnecessary noise. Also not the attempt to draw four separated window panes.

I'll address a few things at a time. First out is the sculpting of the forms. The trees and mountains now look like they have some sort of shape. The Jewel shows off some new bold color adjacencies. The roof texture, while simple, conveys the general idea with only two colors. A roof is mostly a flat plane so a gradient there isn't really suitable.

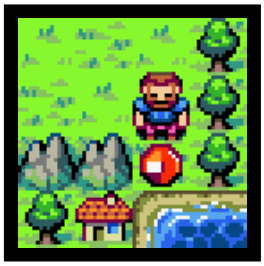
Now I've fiddles a bit with the colors. Note the dark teal in the tree's shadow color, and the warmer, yellowish light green. The mountain has also gotten a cold shadow color and a somewhat warmer light color. The water goes from a deep blue to a teal color. However, these colors are still a bit too "local" (grass is green, water is blue, rock is gray etc.).

Time to throw in some less saturated hues to play against the otherwise overwhelming saturation. I've added some beach sand, but made it a gray-green yellow instead of cartoony yellow. Sand gets darker when wet, so I also suggested that. The mountain now has several colors and is no longer a chunk of gray resting on the grass.

Actually, the mountain did look like it was almost floating, didn't it? I generally try to not do black lines underneath shapes which have ground contact. Here I've removed the lines under the mountain, tree, house and guy. It's possible to skip outlines all together and simply rely on color contrast to separate things, but I'll stick with the outlines for my fix here.

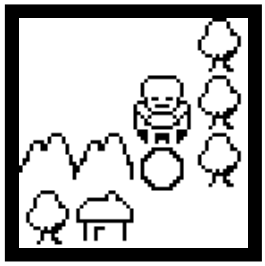


Internal black lines can take up unnecessary pixel space (external outlines too) and tend to gray the adjacent colors (it's like mixing in black). Getting rid of the internal lines gives our guy a bit more silhouette, but he still has a lot of internal details creating a mess. Note that internal black lines are by no means "forbidden", they can be used to separate things which need separation (such as the head from the torso in this front+top view). Stuff like the shirt and pants can simply be separated by color difference. Other areas can be separated by color contrast, such as his hairline.



Whoa! What happened here?

Simplification and structure happened. First I cleaned up the black outlines, removing jaggies (pixels which make a line look jagged and rough).



I then chose to lighten up the upmost black lines, especially those which surround a lighter color. I like to do this because the color of the line rubs off on the surrounded color, making it more pleasant, and with a light source coming from the top, it sort of makes sense to only use black for the lower (shadowed) outlines. A full black line works too, especially if you're working with a limited 4 colors per character palette, however, for shiny stuff like suns and plasma blasts I'd go for a fully colored outline (if any). Fluffy materials might also have a softer colored outline. For some type of gun barrels, I omit the outline to suggest openness.

The grass was a noisy mess and needed some structure. One approach is to make little islands of detail. Imagine the pixels gravitating towards each other to form little patches/clusters. Then, grass structure had to be conveyed, so I chose a dark green to suggest individual strands popping up.Grass is flat in games and shouldn't interfere with the characters treading upon it (like the noisy earlier version did).

Doing two or more different grass tiles is a good idea (a busy tile and a more empty tile). This allows the larger plains to look a bit less uniform and this having some areas of interest. The ocean received a similar treatment, but different structure. Before, the grass and ocean had the same structure (noise).

The little darker vertical lines on the window and door helps to give those areas color and separation from the wall (which would be harder with just a flat light brown and light blue).

Finally, I fiddled around a bit with the character, moving down the face to suggest a front-top view. I brought back some of the internal black lines for the blue armour because it's rather dark and it also has more forms likely to cast shadows and cause overlap. A thin white t-shirt would've been a different deal.

Of course, when I tried out the tiling on a larger scale, it didn't look very good. Tiling is one of those things which has to be solved by trying out the stuff in-game. My grass still looked very noisy/dotty when filling half a screen, so I made a very quiet version and a more busy version.

The mountains were a different story. Mountains are very hard to do in 16px tile engines like this, because they are either squares (such as Zelda 2) or cones sticking out of the grass at even intervals (Dragon Quest 1). Here I tried making a version with a little grass and one that's more square and it seemed to work the best (some failures can be seen below). I added some texture to the greenish base of the mountains to suggest vegetation rather than some form of discolor.

It helps to use rhyming shapes. The alternative versions need to look different, but the same! Note the forms on the trees.

The beaches got a little dithering on the shadow side, doing the double duty thing I mentioned earlier (texture and shadow combined). I caved in and gave the house some front-top perspective. This meant widening the chimney so I could put in a perspective dot. In engine..

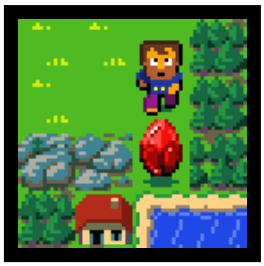
I added some fogged stuff for enemy encounters.

Second opinion

I invited a few other pixel artists to do something with the image because I felt that my own attempt lacked range, i.e. I didn't want to suggest that my way is the only way to fix up the original image. My attempt also stayed rather close to the original.

Ryumaru > Quickly became unsatisfied with this but thought I'd show anyway. Perhaps at least it fixes some of the issues :p Gradients and unnecessary colors were prevalent in the original, that was the first issue I addressed with each asset. Instead I focused on tightly placed pixels and kept a light source direction in mind to convey form in the small area given. Focus was placed on how different surface materials reflect light to better convey them. Single pixel noise was minimized to allow the eye rest in areas that would have large expanses such as the grass and water.

YellowLime > Anything I have to say is mostly a repeat of Ryumaru's... I reduced the amount of colors (maybe way too much) and created assets in a 3/4 perspective instead of the previous flat look. Obviously I did without the (irregular) black lines. Overall I went for a simplistic look and didn't go into detail, both to save time and because I probably wouldn't pull it off :lol:



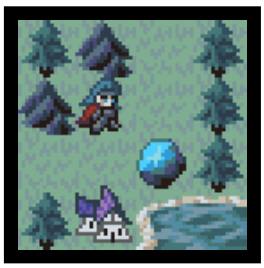
Fickludd > I focused on removing unnecessary colors, black lines and single pixel noise (although the water speculars were kinda nice =)). Tile colors were desaturated compared to sprites. Duplicated tiles were used to make double versions to try to break the grid.

Rosse > I don't care about color-count (36 up to 43). Because the tile are so small, clarity is most important. Each tile has its own sub-palette, no reuse of colors. Increases readability through color-coding. For the pixel-clusters: I use mainly flats, tried to avoid gradients and single-pixel noise. No black outlines or lines inside forms to separate different parts. The design of the objects are symbols, abstracted forms which should convey a general idea, not something specific. The perspective changed from side-view to somewhat top-down-front-ish. Light comes from above, object cast shadows underneath them to increase the feel of 3d space. The character is 23px out of 24px high, so it can bob 1px while walking.

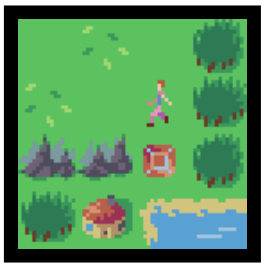
Decroded > Removed all noise and kept contrast limited to points of interest such as the edges of game space. Added a red cape to the character to help him pop. Character probably still needs more brightness but I'm out of time. * concerned that grass is too noisy*

Jeremy > I'm inexperienced with tiling, so I've probably messed up the water edge ones somewhere :yell: I tried to keep bubblegum sorts of colours, just less eye-burning than the original. Also simplify, remove single-pixel noise, etc.

LetMeThink > Here's my half hour effort. What I did: CLEANED UP THE OUTLINES. DESATURATED THE COLOURS. Chose a light source. Drew in form. Made some bad grass. Adjusted hues slightly. Made some awful water. Decided that 1 gem wasn't enough, so made 3. Added some shadows. Made an awful effort at a character as i cannot draw them Tidied things up slightly.

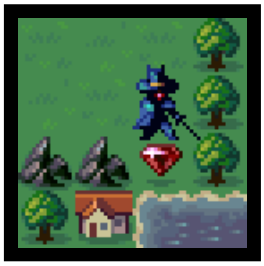


Pix3M > Generally, I made the shapes and forms more interesting. I also went for something that could look like you're in a mystical, dangerous world. I went with pines trees because they're great for environments that is pretty much apathetic to you, while deciduous trees are more extroverted and open. I changed the house from being a plain square house to something a little more unconventional. Possible directions to take this farther are adding cast shadows and polishing up the colors, possibly bouncelight, maybe.



MrFahrenheit > Messed around with shapes and colors. It would've been up sooner but my computer lost power and I hadn't saved... :D . I had fun doing it at least!

DawnBringer > Couldn't resist playing with that mini stuff. Stole all your best designs and tweaked them! ;) Lottsa colors, mostly due to separate palettes for most items. Iso house! works ok me thinks, right? Character is 1 pixel too wide...didn't wanna compromise it :/



Wolfenoctis > Had fun doing this

Yaomon17 > arrr, didn't do the edges of the water. No time to change it now... Tried to use less saturated colors. I just like less saturated colors.

Some alternative retro versions

Graphics on computers such as the ZX Spectrum had a very flat look and black game backgrounds were very common. When you can't use the colors you want or do fancy sculpting of volumes using light and dark colors, you're almost forced to use iconic/cartoony representations of objects (when dealing with smaller sprites at least).

Gameboy plains were sometimes completely barren (flat color) but here I've opted for also adding some triangle formation (often tiles well) grass tufts. I had some fun with a tilted perspective on cliffs and trees and chose to make the trees darker and cliffs lighter to differentiate them. The house got a more original hamburger look (in a fantasy game, we don't want dull normal houses, right?). The Gameboy character sprites only had 3 colors because one was used for transparency, but this is not a big problem when not working in color. Only color intensity (value) is important and you can do a lot with just 3 values.

Hardware restrictions

Undesirable states far outnumber desirable states. You can't just throw paint around and end up with a great painting (Pollock might object). Finding a desirable state of a painting often means that you have to consider certain restrictions, such as how humans look, how light works, and more abstract stuff like our brain's obsession with structure, symmetry and patterns. Harsh hardware restrictions can make it a difficult task to get things to look right, but I've found that the restrictions can also somewhat converge with those of analog art where color count and resolution is much less of an issue. Reducing the amount of colors and unifying the palette is often quite effective, as is removing gritty distracting details.

>>> Move on to discussing the various old palettes, sprite limits etc. Cover the benefits of certain palettes and the benefit of a palette in general. Maybe I'm already doing this above?

More!

- Add my Batman NES analysis somewhere here.
- My 16 color palette.
- Write something about low key and high key areas. All mid key can look flat.
- Talk about my color cube histogram tool?
- Talk about my old sparse sample point down-sampler and auto rotation tool?

Style analysis?

>>> This tutorial is kind of narrow in style (I keep doing my thing), so it would be nice to take a look at different modern styles such as Cave Story, Box humans with 1px legs, etc. Maybe fighting games. There's also games like Raiden II which has graphics which are made to be rotated. Talk a bit about how non-noisy rotation can be made.

Art by Arne Niklas Jansson

AndroidArts.com

Resources and Support

Topic: The Pixel Art Tutorial (Topic Closed)

Author | Message

cure
Commander

Joined: 30 July 2021
Online Status: Offline
Posts: 2859

Topic: The Pixel Art Tutorial
Posted: 27 November 2010 at 10:04pm

Purpose:
This tutorial is designed to explain what pixel art is, what pixel art isn't, how to get started making pixel art and how to make your pixel art better. It is an attempt to consolidate the information scattered throughout the "noobtorials" thread and elsewhere. For more advanced information on what makes pixel art tick, the reader is advised to read the less general tutorials found elsewhere, as well as the Ramblethread! found over at Pixelation, which offers a more in-depth analysis of pixel clusters, banding, and anti-aliasing, and is the source of much of the information found in this tutorial.

Table of contents:

III. Terms to know

1. Anti-aliasing (AA)
2. Dithering
3. Pixel clusters

IV. Things to avoid

1. Bad AA
2. Jaggies
3. Bad dithering
4. Banding
5. Pillow-shading
6. Noise
7. Sel-out

V. Creating a palette

1. When should I worry about colors?
2. Color counts
3. Saturation, hue, luminosity
4. Hue-shifting
5. Color ramps

Edited by jalonso - 28 July 2014 at 6:31am

IP Logged

cure
Commander

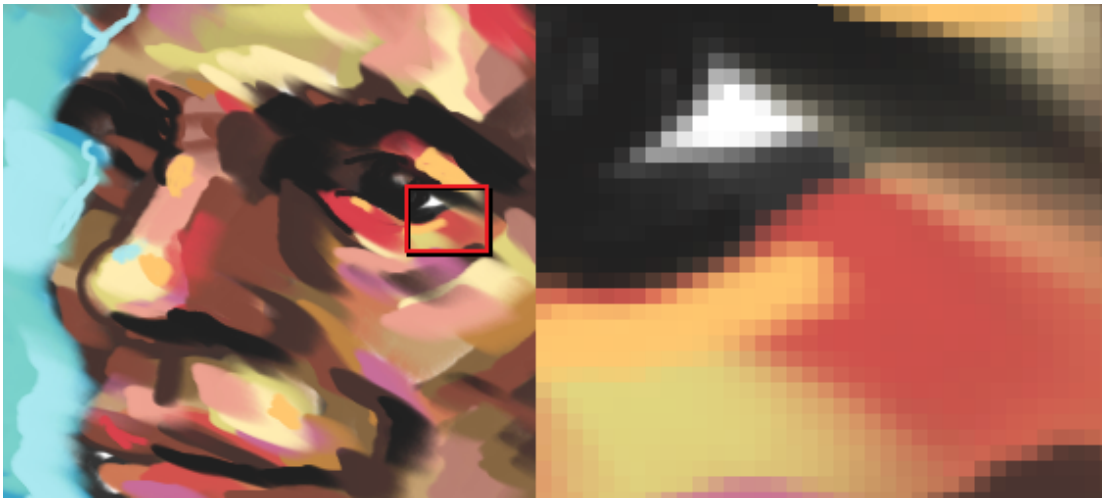
Joined: 30 July 2021
Online Status: Offline
Posts: 2859

Posted: 27 November 2010 at 10:22pm

I. What is Pixel art?

Judging by the name, we might assume that pixel art is any art that's made up of pixels. But not every digital image is pixel art.
This photograph is made from pixels, but is not pixel art:

Alright, so no photographs. But if I make my art on the computer, then it's pixel art, right?
No. Pixel art is a very specific sub-category of digital art. It isn't what it's made of so much as how it's made.
For example, this digital painting is art made on the computer, and it is made of pixels, but it is not pixel art:



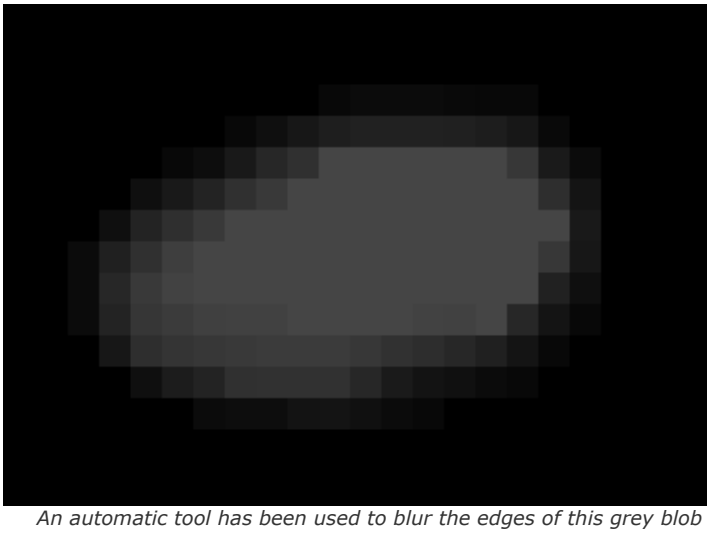
If the pixel art loses the sense of the importance of the pixels which construct it, then I don't think it can be called pixel art. It is when the pixels hold importance to the nature of the work which defines it as pixelart. - Alex HW

Why not all digital art is pixel art

Pixel art is set apart from other digital art forms by its focus on control and precision. The artist has to be in control of the image at the level of the single pixel, and every pixel should be purposefully placed.
When pixel art is done purposefully, offsetting just a few pixels can have a dramatic effect on the image:

The features of this parrot change drastically, but only a few pixels are different.

Other digital art forms use many tools you won't find in pixel art. The reason pixel artists don't use these tools is because they place pixels in a manner that the artist can't predict. These **automatic tools** blur, smudge, smear or blend the pixels. Any tool that places pixels automatically (which means the computer makes decisions about the placement of pixels rather than the artist), is generally frowned upon in pixel art. Remember, pixel art is all about control.



An automatic tool has been used to blur the edges of this grey blob

You'll often hear people complaining "*This isn't pixel art, it has too many colors*!" This isn't because there's some unwritten rule in pixel art that says "It's only pixel art if it has [X] number of colors", you're allowed to use as many colors as you want. The main reason that people complain about color count is that a high amount of colors can indicate the use of dirty tools. Dirty tools create a lot of new colors in order to achieve their blurring, smudging, or transparency effects. People also mention high color counts because larger palettes are more difficult to control, but we'll get to that later.

Why it's not just about the tools

So if I don't use any blur effects or filters or fancy tools, it's pixel art, right? Anything made in MS Paint will be pixel art?
No. It's not the program that determines whether or not it's pixel art, it's how it is made. For example, this image was made in MS Paint, without any fancy tools:

But it isn't pixel art. This is what we call **oekaki**. If you can create the image without zooming in, chances are it isn't pixel art. If you're using the line tool and flood-fill most of the time, you're not paying attention to the individual pixels, just the lines and shapes that the pixels make up. The same goes for rough sketches made with the pencil or brush tools. These methods ignore the importance of careful, deliberate placement of the individual pixels.

While the most common misconceptions about pixel art are due to too loose of an

interpretation of the medium, there are some who have too strict a definition of what makes pixel art.

Every pixel does not literally need to be placed by hand
The job of the pixel artist is not to manually place each and every pixel. You aren't expected to behave like a robot, filling in large areas with thousands of single-clicks of the pencil tool. The **bucket tool** is fine. The **line tool** is fine. What's important is that the artist has control of the image *at the level of the single pixel*, not that you create the image one pixel at a time.

Edited by jalonso - 28 July 2014 at 6:35am

IP Logged

cure
Commander

Joined: 30 July 2021
Online Status: Offline
Posts: 2859

Posted: 27 November 2010 at 10:38pm

II. Where do I start?

Pixel art is about the pixels- that's as simple as it gets. These tips share a common goal: to make sure your focus is on the pixels.

Start small- The larger the image you're trying to make, the more time and work it's going to take to complete it. Don't make this tough on yourself, use a small canvas. Pixel art can convey a lot of information for its size, you'd be surprised how little room you need if you control the pixels properly.

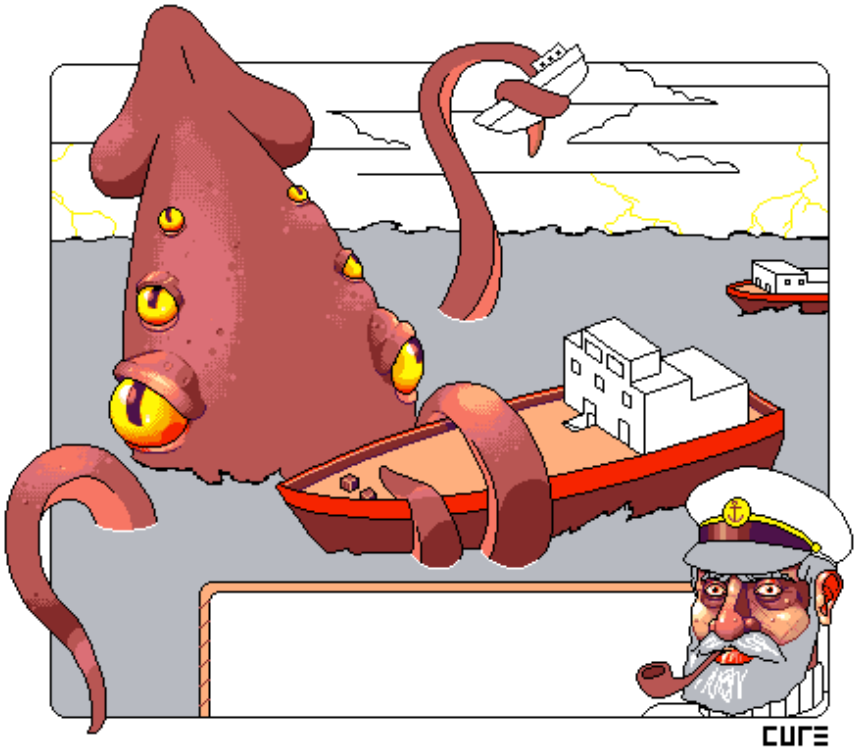
Use a limited palette- If you can't make a good sprite in 4 colors, using 40 colors isn't going to help. Using a small palette is especially good for beginners because it forces you to focus on pixel placement and the relationships between groups of pixels.
The original, 4-color GameBoy palette is a good choice for beginners, as you'll only have to worry about value, and not hue or saturation.

Programs
There are plenty of good programs out there for pixel art, many of which are free. I use Grafx2, but GraphicsGale, Pro Motion, Photoshop, Pixen, and MS Paint are all common choices. Some are more user friendly than others, which is why I choose something with keyboard shortcuts like Grafx2 over MS Paint, it has saved me many trips to the toolbar (and makes for much easier palette management).

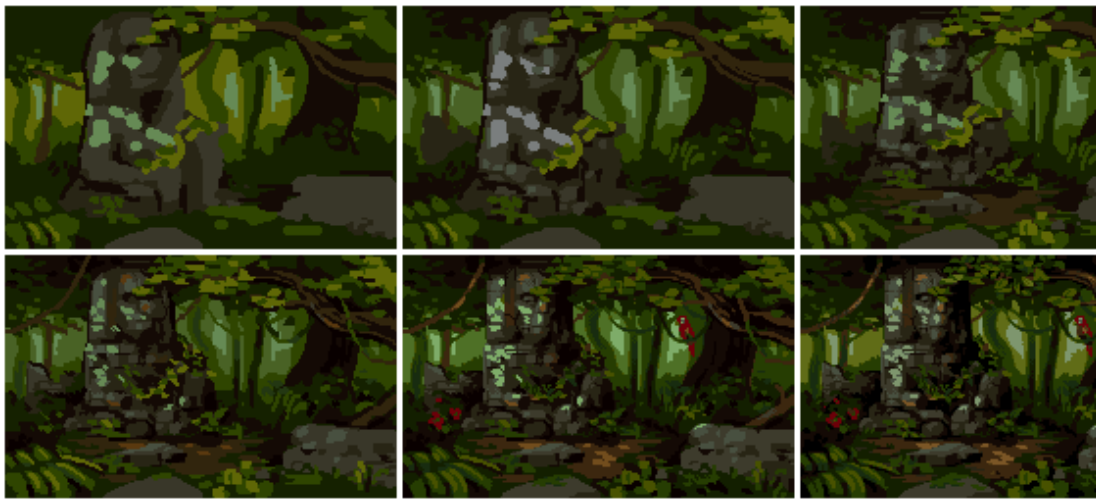
File type
A common mistake that new pixel artists make is saving their art as a JPEG/JPG. While this file type might be fine for other types of images, it causes compression, which destroys the quality of a piece of pixel art.

Never, ever save as JPG. Instead, save as PNG or GIF. Be careful though, as some programs (such as MS Paint) don't properly support the GIF format, and will ruin your image. In these instances, you'll need a file converter (such as Giffy) if you want to save your image as a GIF.

But how do I *start* the image?
It's completely up to you. Some artists prefer to create the line art first, then go in and add color:



Other artist prefer to 'block-in' the major forms with a larger brush, then continue by refining the image until it has a pixel-level polish:



Both methods are fine, it all depends on what you're comfortable with, or the specifics of the project. Line work might be a good method if you're tracing a scanned image (such was the case for the sea monster example above). If you're beginning the image in your pixelling program, and it isn't a tiny sprite, blocking in the forms with a larger brush may prove more useful.

Edited by jalonso - 28 July 2014 at 6:37am

IP Logged

cure
Commander

Joined: 30 July 2021
Online Status: Offline
Posts: 2859

Posted: 27 November 2010 at 10:50pm

III. Terms to know

Anti-aliasing (AA):

[In addition to the information found in this section, check out this image by Ptoing.]

Anti-aliasing is the method of making jagged edges look smooth. You may be familiar with anti-aliasing already, because a lot of programs and tools do this automatically. When we're talking about pixel art, however, anti-aliasing means **manual anti-aliasing**. Manual AA means smoothing the jagged areas by hand-placing pixels of a different color to ease the

transition. Here's an example:

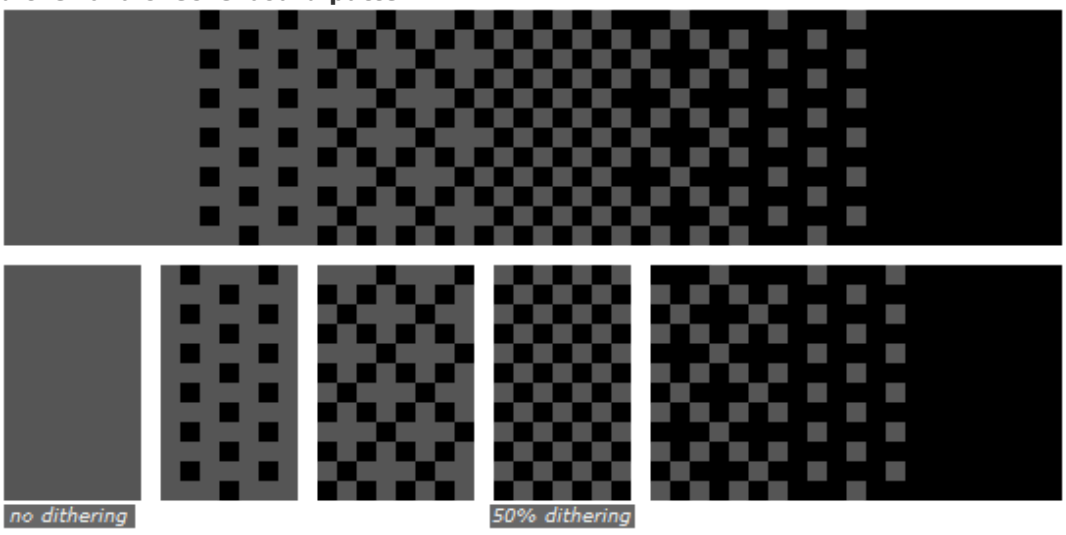
without AA *with AA added*

There are several pitfalls often encountered when applying anti-aliasing, which are discussed in the "Things to avoid" section.

Dithering:

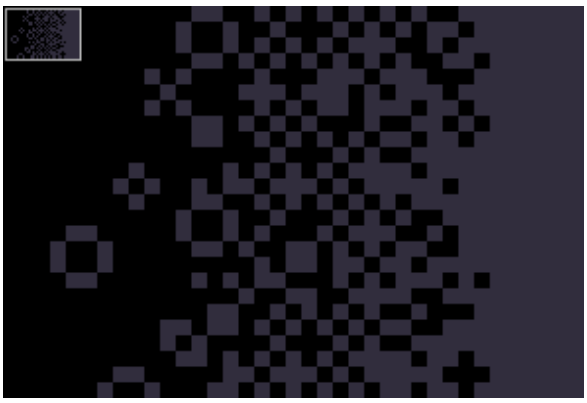
Dithering consists of different patterns of pixels. It's typically used to ease the transition between two colors, without adding any new colors to the palette. It's also used for creating texture. In the days of CRT monitors, dithering was especially useful as the screen would actually blur the dithered area and obscure the pattern. Now that crisp LCD monitors are the norm, the patterns are no longer as easy to hide, meaning dithering is not as versatile as it once was. Even so, dithering still has its uses.

The most common form of dithering you'll see is a **50/50 dither**, also known as a **50% dither** or a **checkerboard pattern**.

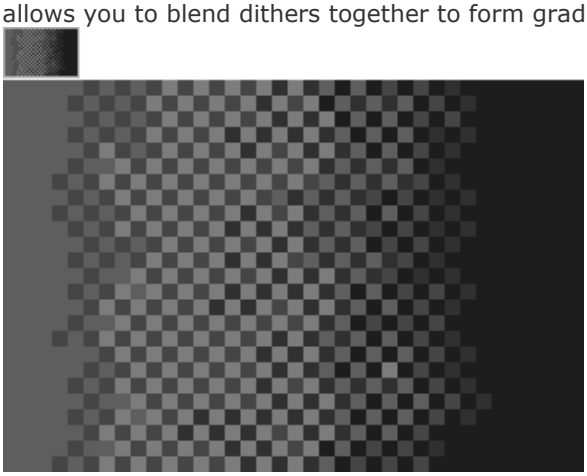


As shown in the example above, you can create various other patterns to further buffer between a full color and a 50% dithering pattern.
These patterns are often easier to spot than a 50% dither though, so be careful!

Stylized dithering is another technique, and is characterized by the addition of small shapes in the pattern.



Interlaced dithering allows for two dither regions to hug each other. It is called interlaced dithering because the two dithers weave together at the borders. This type of dithering allows you to blend dithers together to form gradients.



Random dithering is a less-common form of dithering, and isn't generally advised, as it adds a lot of single-pixel noise to the image. While it has some usage in very small doses, random dithering is something you'll often want to avoid.

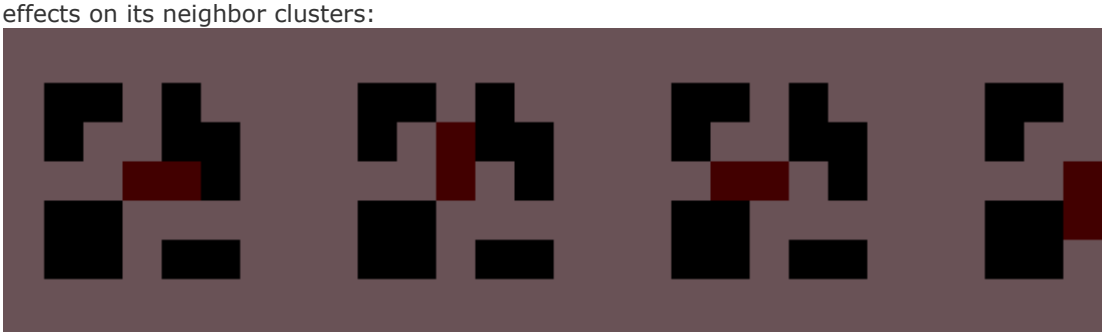
As useful as dithering is, it's often misused by inexperienced artists. **Bad dithering** is discussed further in the Things to avoid section.

Pixel Clusters:

The cluster of pixels is made from single pixels. However, a single pixel is most of the time near-useless and meaningless if not touching pixels of the same color.
The pixel artist is concerned with the shapes that occur when pixels of similar color touch each other and convey an opaque, flat, shape.
Most of the defeats and possible triumphs of pixel art occur in that exact moment where the artist makes a cluster of pixels.
-Ramblethread

I stress the importance of placing individual pixels, but these are rarely independent pixels. A single pixel, isolated, is a speck on a screen- it’s noise. But pixels aren’t usually found alone, instead they exist as part of **pixel clusters**- groups of pixels of the same color that together produce a solid color field. While the single pixel is our basic building block and smallest unit, the pixel cluster is the unit on which much of our decisions about pixel-

placement will be based. And while it's important to realize individual pixels aren't independent, it's just as important to realize pixel clusters aren't independent. Like puzzle pieces, the borders of a pixel cluster determine the shape of the pixel clusters it borders. Here is an example of how rearranging the shape of a pixel cluster can have dramatic effects on its neighbor clusters:



While lone pixels often read as noise, a lone pixel of a color different than the field it touches, *if used as a buffer* (AA), reads as part of that cluster, and is thus unproblematic:

Edited by jalonso - 28 July 2014 at 6:53am

IP Logged

cure
Commander

Joined: 30 July 2021
Online Status: Offline
Posts: 2859

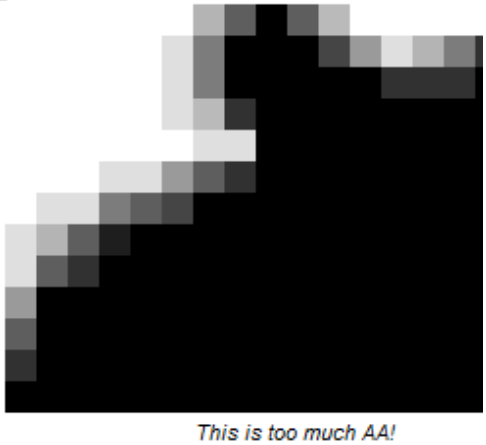
Posted: 28 November 2010 at 12:01am

IV. Things to avoid

Bad AA:

Too much AA (over-anti-aliasing)- You only want to use as much AA as is necessary to smooth the edge. If you use too much, the edges can look blurry, and you lose the crispness of the line.

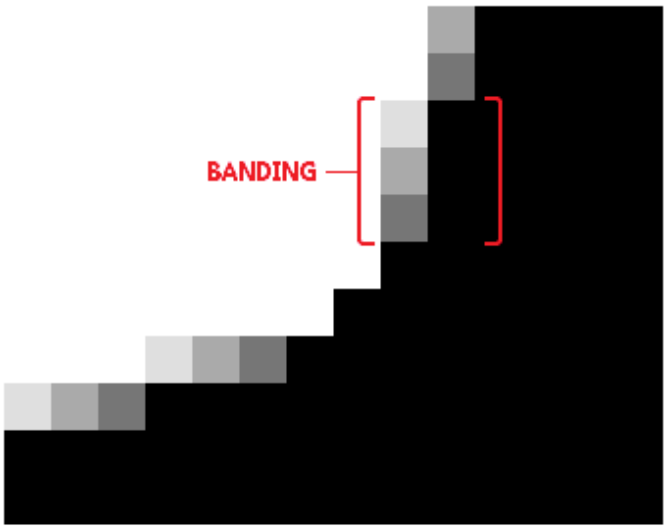
The AA smooths the edges but doesn't add blur



This is too much AA!

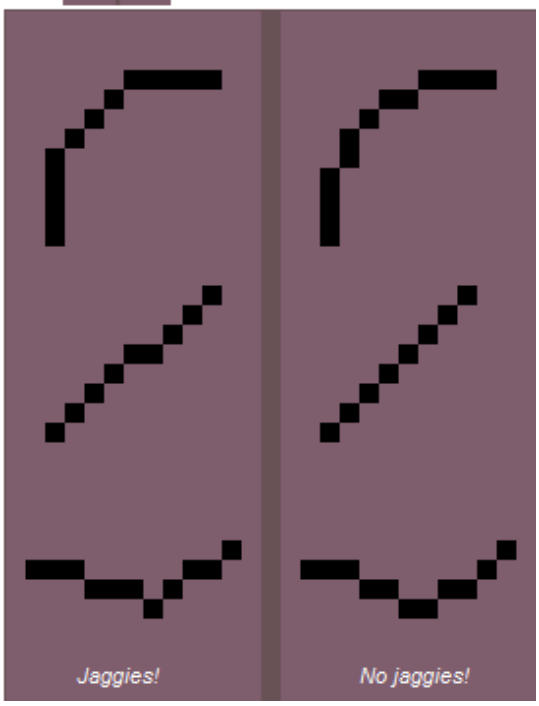
Too little AA- Here the artist has used single pixels to ease the transition, but he has only succeeded in blunting the jagged edge a bit. He could have made a much smoother transition by using longer lines of pixels to show a more gradual transition:

AA banding- When segments of AA line up with the lines they're buffering, AA banding occurs. For a better understanding of AA banding, be sure to read the section on banding.



Jaggies:

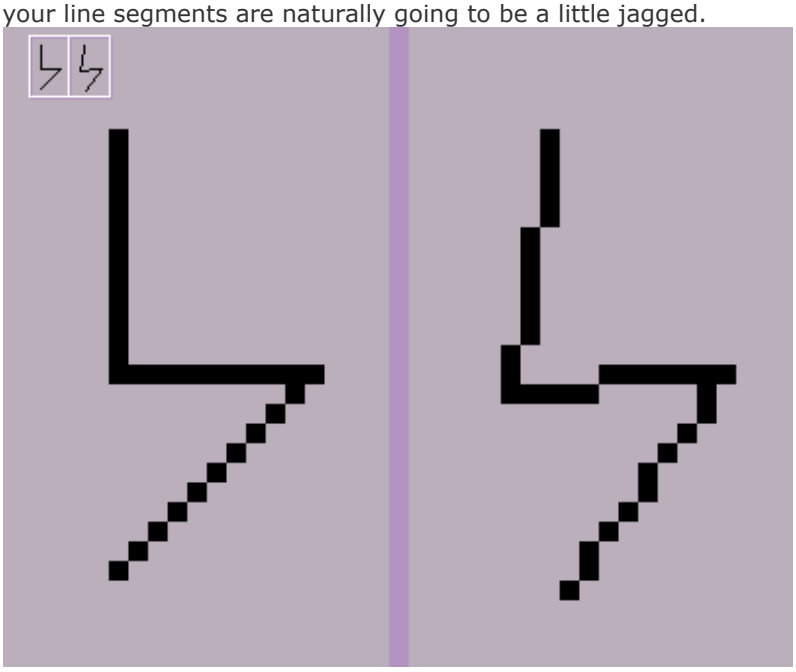
Jaggies occur when a pixel or group of pixels are out of place, interrupting the flow of a line. Jaggies can also occur when a line lacks anti-aliasing. Jaggies get their name from the jagged lines that they create. More broadly, jaggies are the result of any bad pixel technique, but they are most often discussed in reference to line work, so that is the context in which they will be discussed here.
How to fix jaggies:
Changing the length of the lines



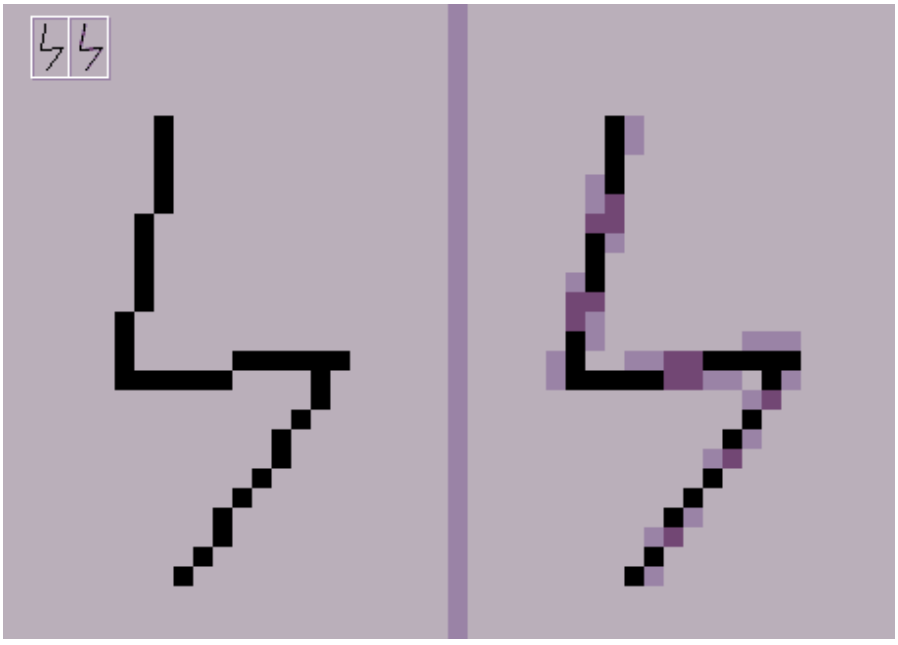
often times the problem is just that a segment of the line is too short or too long, and it creates an awkward jump. Using a more uniform length of pixels to smooth the transition is the solution here.

Anti-aliasing

Unless your line is perfectly horizontal, perfectly vertical, or at 45 degrees, the edges of your line segments are naturally going to be a little jagged.

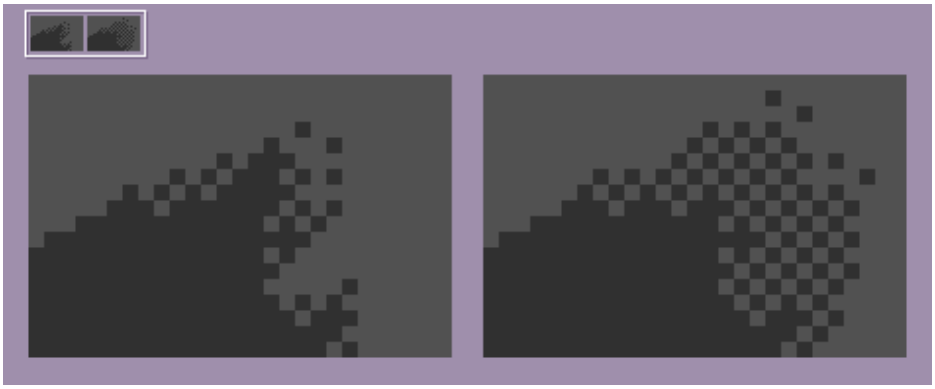


This is because the square nature of the pixel and the grid pattern we're restricted to makes angled lines and curves difficult to portray. AA is the correct counter-measure in these situations.



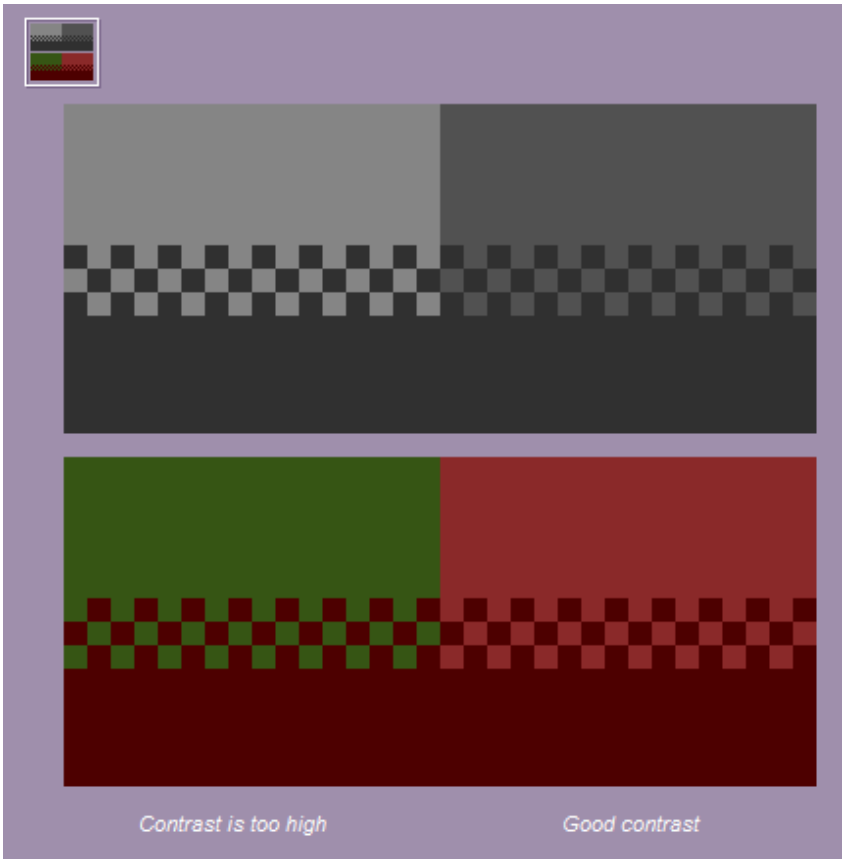
Bad dithering:

There are several common ways dithering is misused. The most common mistake is simply using **too much dithering**. If dithering is covering half your sprite, it'd probably just be better if you added a new color to the palette. Dithering should ideally be used to taper the ends and edges of an opaque field of pixels. When too much dithering is used, the dithered area turns into a field itself:



At this point dithering is no longer serving as a buffer between colors, but creating unwanted texture. Creating texture can be a useful aspect of dithering, but only when used correctly. If you're trying to buffer and are instead adding texture, then dithering isn't working out.

So how much dithering should you use? Well, it depends on how big your palette is really- or more precisely, the contrast between the two colors you're trying to dither with. The lower the contrast is between the two (in hue or in value), the less harsh the dithering will be:



Banding:

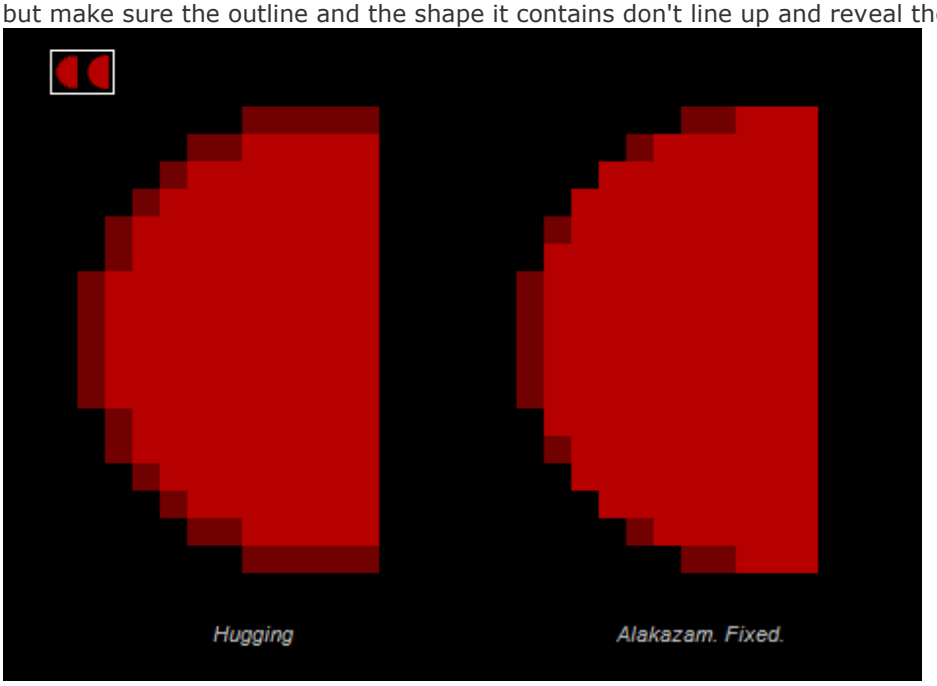
Banding, most simply, is when pixels line up. When neighbor pixels end at the same x or y

coordinate on the underlying grid, the grid immediately becomes more evident, the pixels are exposed, and the apparent resolution becomes less fine.

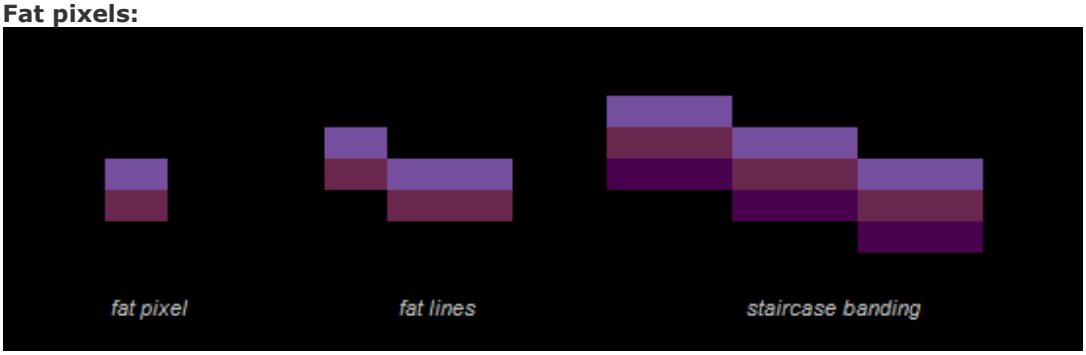
Here are several instances of banding, all of which occur because the pixels have lined up. These names aren't common lingo, but will work for the purposes of this tutorial:

Hugging:

Here an opaque field of color has been outlined by a row of pixels. It's fine to use outlines, but make sure the outline and the shape it contains don't line up and reveal the grid.

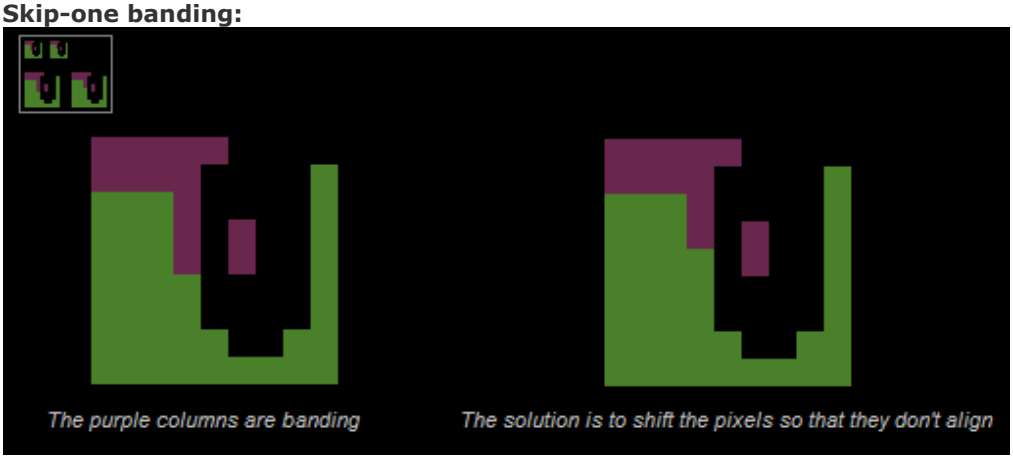


Fat pixels:



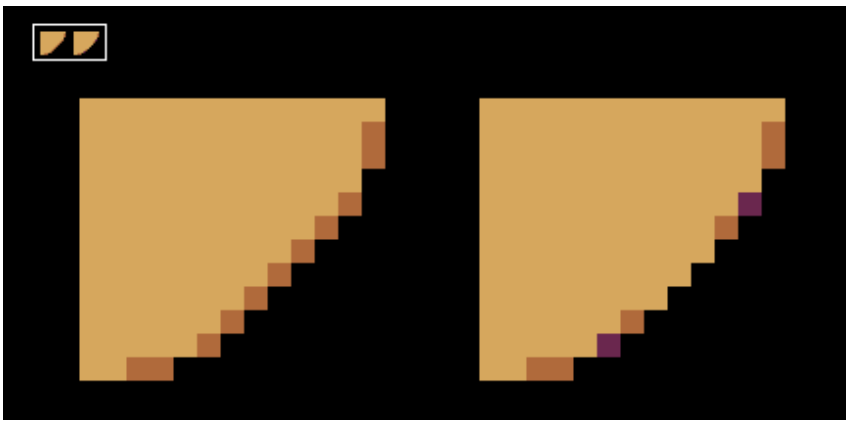
Fat pixels can occur alone in small squares, together as fat lines, or multiplied as large bands (staircase banding).

Skip-one banding:



Even if there is a negative space between two bands, the mind will fill in the gap and banding will remain.

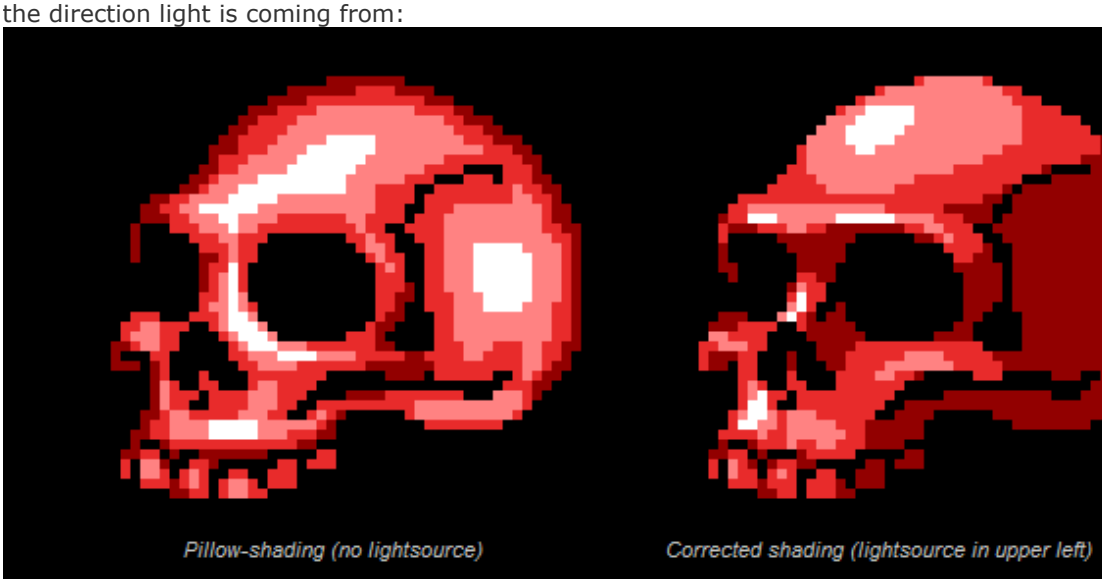
45 degree banding:



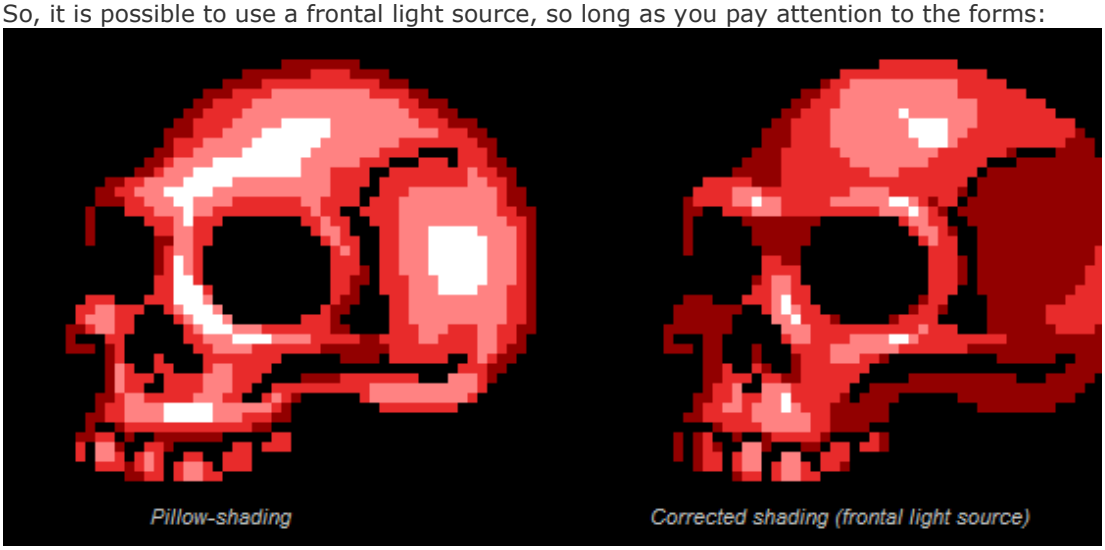
Though the rows of pixels lining up are only 1 pixel thick, banding is still present.

Pillow-shading:

Shading by surrounding a central area with increasingly darker bands. Pillow-shading is bad because it pays no attention to the light source, and conforms to the shape of the area rather than the form it represents of how light affects it. Pillow shading is often, but not always, combined with banding. The way to fix pillow-shading is simply to pay attention to the direction light is coming from:



The reason pillow-shading is wrong is not because the light source is frontal (from the viewer's direction). You don't have to place the light source in the corner. The reason pillow-shading is incorrect is because it follows flat shapes rather than focuses on how the three-dimensional forms are lit.
So, it is possible to use a frontal light source, so long as you pay attention to the forms:



Noise:

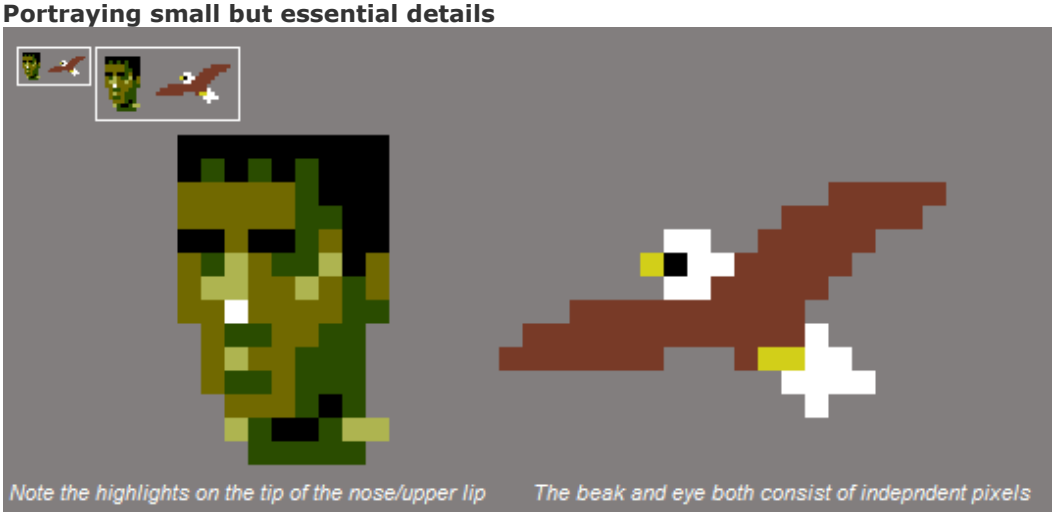
Much of the time, **independent pixels** (pixels that do not belong to a pixel cluster) are unable to convey sufficient information by themselves, and their inclusion usually only creates noise. **Noise** is any sort of information that does not contribute to the piece and serves only to interrupt the area it inhabits and distract the viewer. In pixel art, noise is often composed of independent pixels. For the purposes of this tutorial, single-pixel noise will be what I'm referring to when I use the term "noise". The reason one must be careful when using a 25% dither (or any dithering, really) is because of the noise all the independent pixels create.
Single pixels expose the underlying grid by revealing the resolution of the image. Remember, in the wild, pixels travel in packs. It's the nature of a pixel to long for a place in a pixel cluster. For this reason, independent pixels should only be used for very specific and purposeful reasons.

Justifiable instances of independent pixels include:

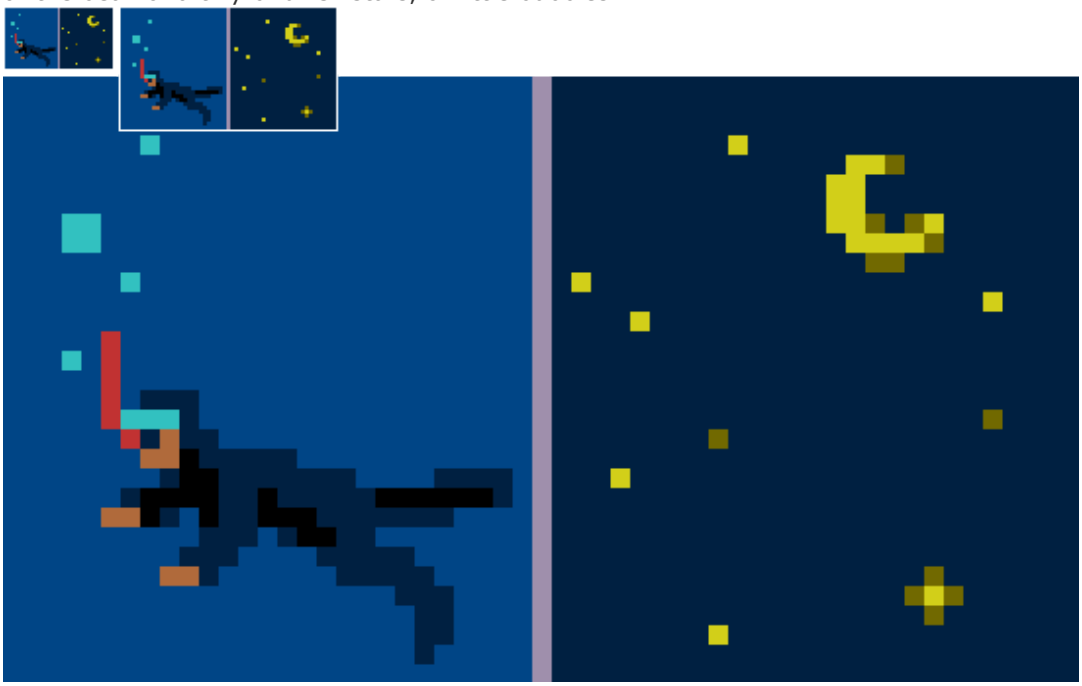
Use as specular highlights

Independent details call a lot of attention to themselves, but sometimes this is precisely what you want. For bright specular highlights, single pixels will often work just fine. For an example, see the white pixel used on the monster's nose below.

Portraying small but essential details



Usually this will only matter for details on very small images, like the eyes on a small sprite, or the beak of a tiny bird. Or stars, or little bubbles.



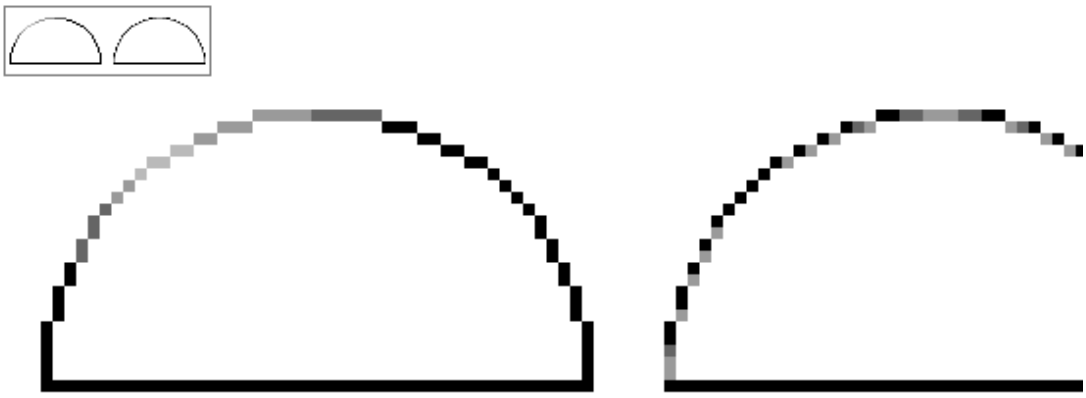
Sel-out (broken outlines)

Sel-out (short for **selective outlining**, also known as **broken outlines**) is **anti-aliasing** an outline to a background color. This means sel-out is really a type of **bad AA**, but the

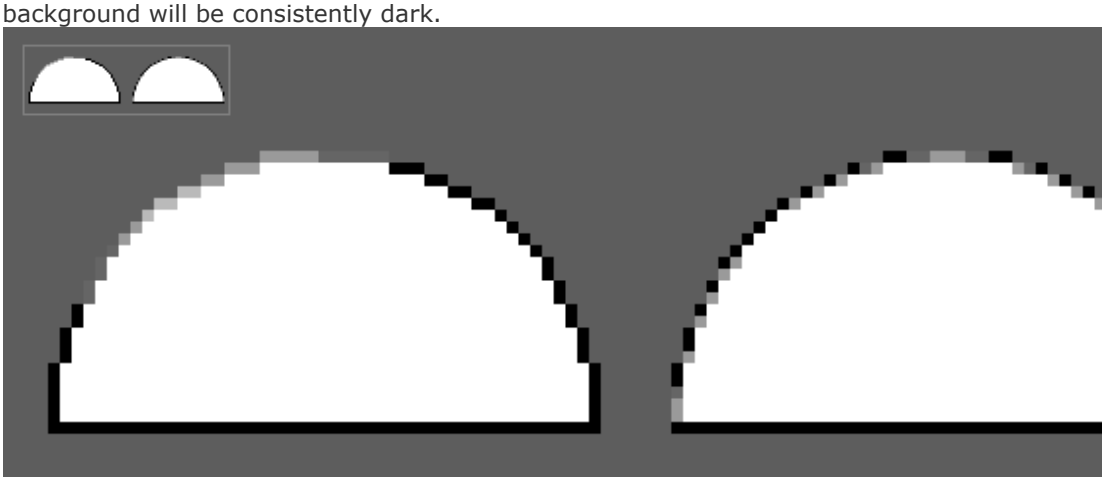
term has become popular enough to warrant its own section.

The idea is usually to darken the outline at the contours to approach a darker color, so that the sprite will read well on any background, instead of melting into a similarly-colored background. Sel-out is **not** shading an outline according to a light source. A full outline with light variation won't create **jaggies** as badly as a broken outline will:

Perhaps this is a simpler example. The half-circle on the left is shaded according to a light source (again, coming from the top left corner). The top of the half-circle on the right has sel-out applied:



Sel-out works if it is created for specific scenarios, such as in a game where you know the background will be consistently dark.



Edited by jalonso - 28 July 2014 at 7:05am

IP Logged

cure
Commander

Joined: 30 July 2021
Online Status: Offline
Posts: 2859

Posted: 28 November 2010 at 11:59pm

V. Creating a palette:

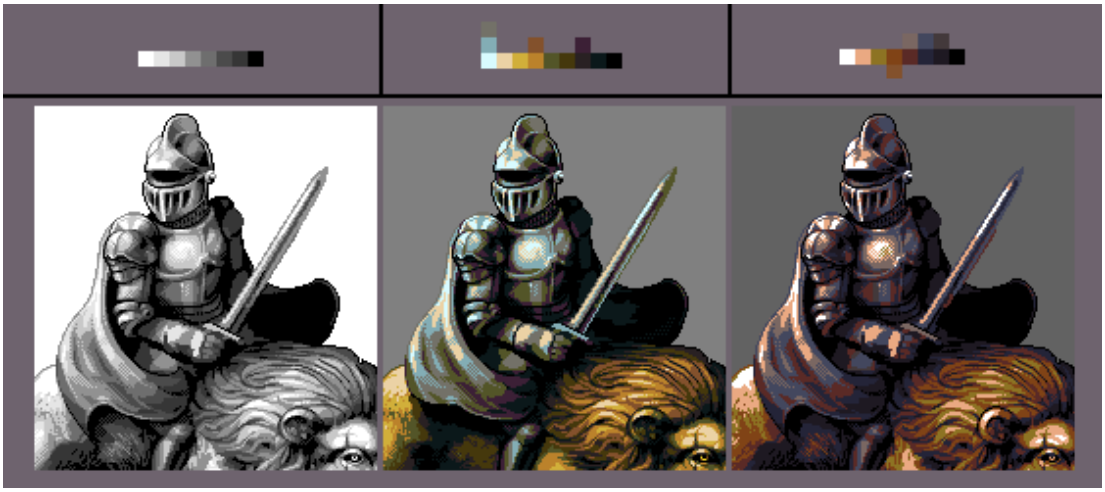
When should I worry about colors?

Well essentially what it comes down to is, what colors does the piece need to have? then, as I go, how far can I get with those (until of course I need to add more shades). That's when the mixing occurs.
-Adarias

This is a common method of creating a palette for a piece. Here's an example of what he's talking about:

As the piece gets more complex, it becomes necessary to create additional colors to achieve more advanced shading, or to color new image elements or details.

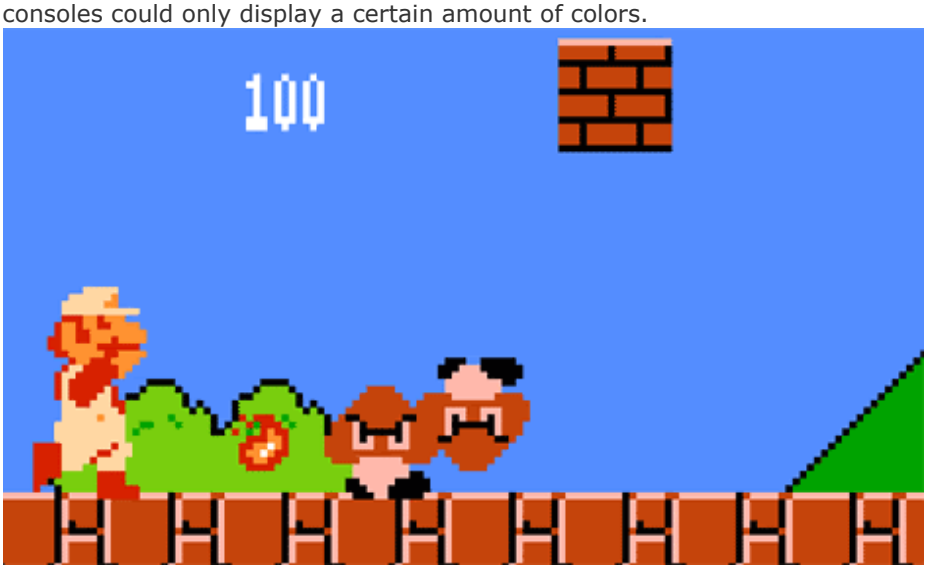
Another method is to create the piece in shades of grey, then add color later. This is possible because relative **value** is a greater concern than **hue**, because hue can be more easily altered later on, after value relationships have been established.



Personally I find it easier to keep up with colors as the piece progresses, so I prefer the first method.

Color count

You may find that pixel artists often advocate a low **color count**. You might assume that this is just a tradition leftover from the olden days of pixel art, back when video game consoles could only display a certain amount of colors.



If modern computers can easily display hundreds of colors, why shouldn't you use them all? In truth, using small palettes isn't an outdated tradition of pixel art, and there are very logical reasons behind this practice.

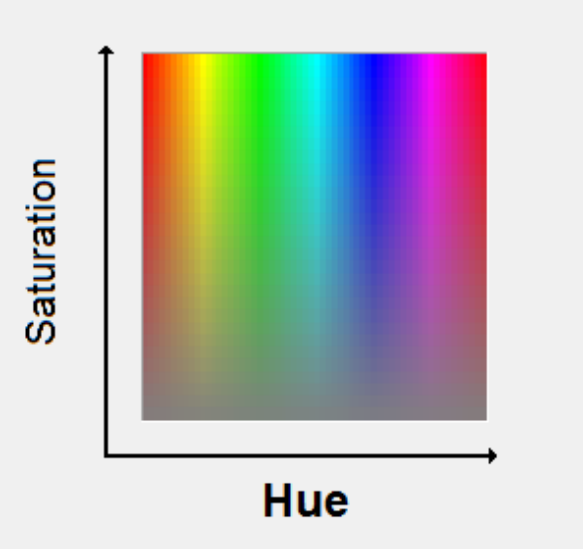
Cohesion- When you're using less colors, the same colors will reappear throughout the piece more frequently. Since the different areas of the work share the same colors, the palette ties the piece together, unifying the work.

Control- The smaller the palette, the easier it is to manage. You may, and probably will, want to change adjust a color later on. If you've got 200 colors, it's going to take you a lot longer to make the adjustments, because by changing one color you've thrown off its relationship with the colors neighboring it on its color ramps, and adjusting them means changing the relationships between those colors and their neighbors! You can see how this quickly adds up to a lot of work. With a smaller palette, the effect of changing a single color is more substantial, and there are less micro-relationships to worry about.

Hue, Saturation, and Luminescence

Hue:

Hue refers to the identity of a color. Whether a color is defined as blue, red, orange, etc. depends on its hue:



In the above picture, hue is represented along the x-axis.

Just as you can change how bright or dark a color appears by surrounding it with lighter or darker pixels, the perceived hue of a color depends on its environment. Here we have a completely neutral, medium grey:

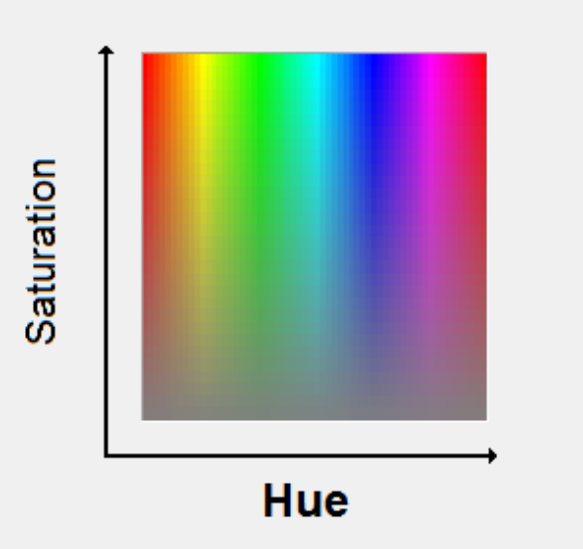
In this picture (a detail of this piece by iLKke) the green in the trees is actually not green at all, but the same grey as the previous picture:

Because the background is so purple (which is the opposite of green on the color wheel), the grey looks greener than it actually is.

Hue will be an important concept later when we discuss hue shifting.

Saturation:

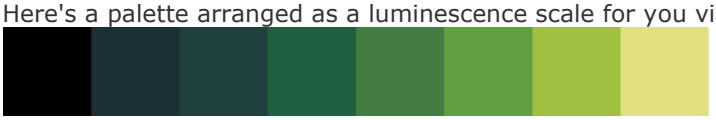
Saturation is the intensity of a color. The lower the saturation of a color, the closer the color gets to grey:



The most common problem new artists encounter is regards to saturation is using colors with too high of a saturation. When this happens, the colors start to burn the eyes. This can be a problem in any media, but because the colors in pixel art are made up of light, instead of pigment as in paint, the potential for colors being too bright or irritating is much higher. Notice how the colors in the second image are much easier on the eyes:

Luminescence (brightness):

Luminescence (also known as **brightness** or **value**) is how dark or light a color is. The higher the luminescence, the closer the color gets to white. If the luminescence is 0, then the color is black.
Here's a palette arranged as a luminescence scale for you visual learners:



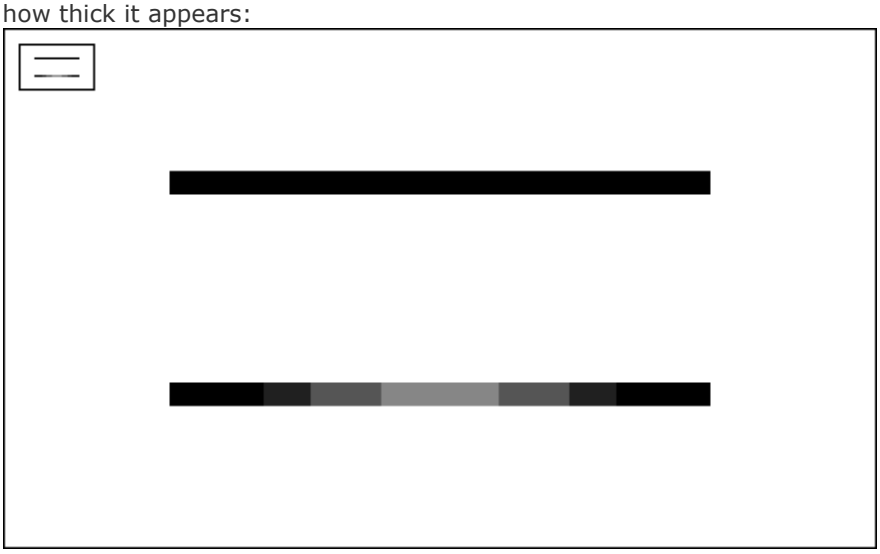
low luminescence (darker colors) on the left, high luminescence (brighter colors) on the right

In a given palette, you'll want to have a wide range of values. If you only have colors in the same range of luminescence, then you won't be able to create good contrast- a full range of values allows you to use highlights, mid-tones, and shadows. The difference between the brightness of two colors is known as **contrast**. A common problem newer artists exhibit is not having enough contrast. Here's an example of an image for which the contrast is too low:

And that same image, adjusted so the values are spread out more evenly from light to dark:

The value of a color is a set number, but colors can appear lighter or darker depending on their background. For this reason, you won't always want to use your brightest color for every highlight. A color that makes a good highlight on one object might be too bright to use on a darker object.

Luminescence is especially relevant to pixel art: The brightness of a pixel or line determines how thick it appears:



The first example is a simple black line. The width of the line looks consistent. Below that is a line with pixels that vary in brightness. Notice how the line appears thinner toward the center at 1x.

Color Ramps

A color ramp is a group of colors that can be used together, arranged according to luminosity. A palette can consist of a single ramp of many different ramps.
Here's a palette:

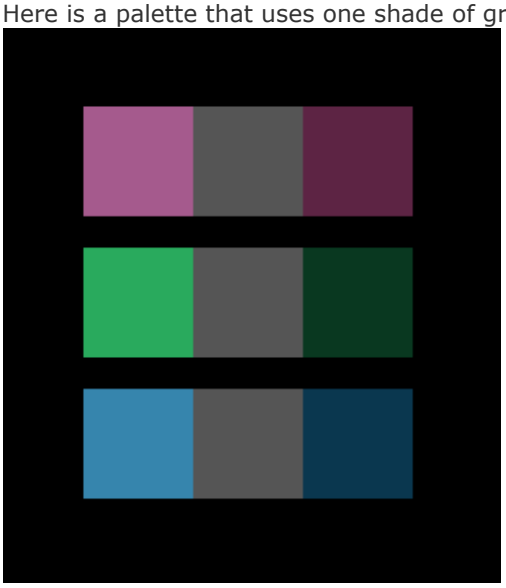
And here's that same palette, arranged according to its color ramps (of which there are two):

It isn’t necessary that you actually create a model like the one above (though some artists find it useful). What is important is that you understand what your color relationships are- that is, what your ramps are.

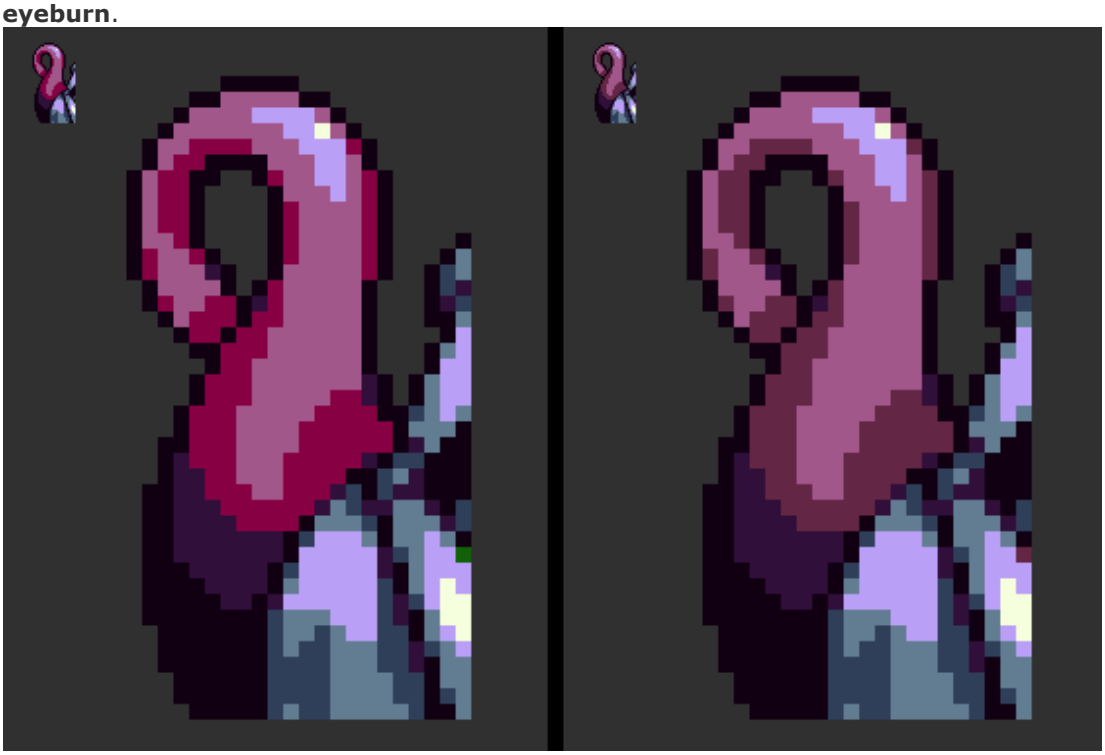
It isn't necessary that a color be restricted to a single ramp. Often, ramps will share colors. Frequently, the darkest or lightest color will belong to most or all of the palette's ramps, as in the example above, in which both ramps share the same darkest and lightest shades.

It’s also possible for mid-tones to work in multiple ramps. In these cases, the versatile color takes the place of two or more separate colors, aiding in palette conservation. In the case of multi-ramp shadows and highlights, the extremes in luminescence allow the color to be flexible (because they approach black or white). Since mid-tones are not afforded this advantage, they are often more neutral colors, meaning they are closer to brown or grey.

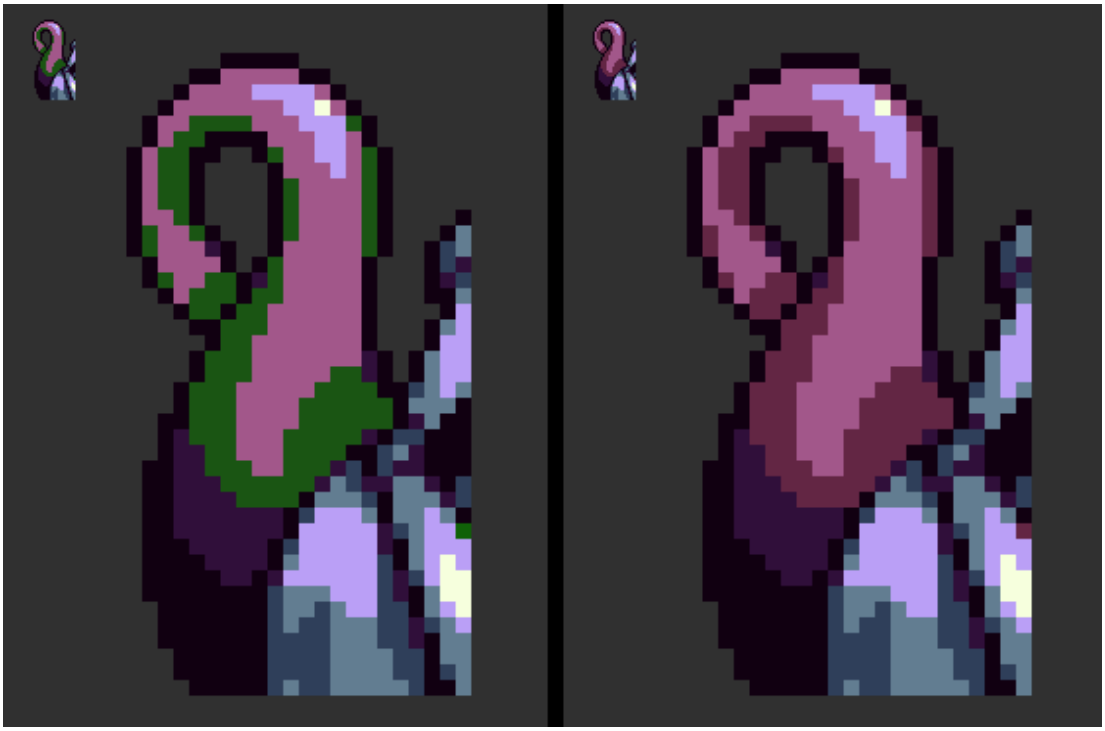
Here is a palette that uses one shade of grey to bridge the gaps in several ramps:



You also have to be careful about having colors in a ramp that don't fit. If a color doesn't belong in the ramp, then it has the potential of **punching through** the image, which is a priority issue in which the color, rather than work as part of the image, seems separate from it, and looks almost like it is sitting on top of the image. This is usually due to the saturation being too high, or because the hue clashes with the neighboring hues, and thus creates **eyeburn**.



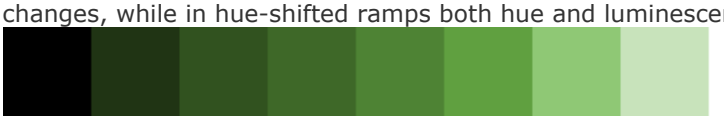
The above image shows eyeburn created by a color with too much saturation.



...and in this image, eyeburn is created by the green clashing with the purple. The hue should logically follow its neighbors in the ramp.

Hue shifting

Hue-shifting refers to having a transition of hues in a color ramp. A color ramp without hue-shifting is known as a **straight ramp**. In straight ramps, only the luminescence changes, while in hue-shifted ramps both hue and luminescence will (usually) change.



The first color ramp is a straight green ramp. The second image is a green ramp with hue-shifting applied. When using hue shifting, bend your highlights toward a certain color (yellow, in the example above), and move the darker colors toward a second color (I chose blue in the above example). Hue-shifting is used because straight ramps are usually boring and don't reflect the variety of hues we see in reality, and hue shifting can add subtle color contrast within a ramp.

Edited by jalonso - 28 July 2014 at 6:49am

IP Logged

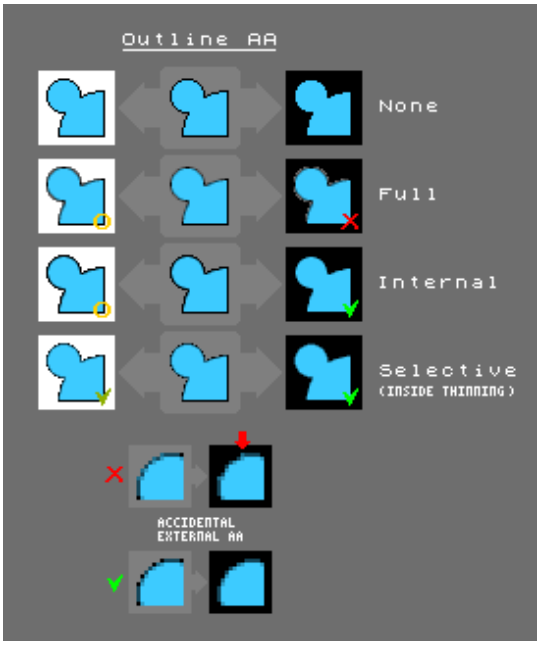
jalonso
Admiral

Joined: 12 August 2021
Online Status: Offline
Posts: 13537

Posted: 28 June 2014 at 9:01am

By DB:
Selective AA is the process of pushing/moving the internal AA into the outline...without ever making the (easy) mistake of doing external AA. "Internal thinning".

If you want strictly pixelart (and no semitransparancy) and have sprites that should look decent on all backgrounds; then "selective AA" is the best method (I don't know if there's an accepted term for it, however it has a close relationship with Selout). Although it's a very advanced pixelart technique, and may not be suitable for beginners..so try working with just internal-AA if you wanna play it safe.



Edited by jalonso - 28 July 2014 at 7:08am

PJs FAQ <•> Sticky Reads

IP Logged

jalonso
Admiral

Joined: 12 August 2021
Online Status: Offline
Posts: 13537

Posted: 20 July 2014 at 6:48am

Link to waybackmacine's backup of the original SEL-OUT tutorial
https://web.archive.org/web/20130602163317/http://pixel-zone.rpgdx.net/shtml/tut-selout.shtml

~~*If anyone has a way to backup this tutorial PM me~~
~~By this I mean copy all pages, images in a way that requires little but uploading to a server.~~

*Tutorial collected by Uncle_B . I'll post asap.

Edited by jalonso - 15 January 2015 at 6:31am

PJs FAQ <•> Sticky Reads

IP Logged

POST REPLY NEW TOPIC

Printable version

Forum Jump -- Select Forum --

You **cannot** post new topics in this forum
You **cannot** reply to topics in this forum
You **cannot** delete your posts in this forum
You **cannot** edit your posts in this forum
You **cannot** create polls in this forum
You **cannot** vote in polls in this forum

MAKE GAMES WITH DEREK.

PIXEL ART TUTORIAL: BASICS

In this fast-paced tutorial, I show you the basics of making pixel art by walking you through the creation of a **sprite**. Sprites are the images in 2d games that represent the various objects in a game like your player character, monsters, items, etc.

This tutorial is paired with a follow-up tutorial called PIXEL ART: COMMON MISTAKES.

QUICK LINKS

BACKGROUND

Metal Slug 3 (Arcade). SNK, 2000.

Pixel art, also known as **dot art** in Japan, is a form of digital art where editing is done on the pixel level. It's primarily associated with the graphics of 80s and 90s video games, where commercial artists strained against limited memory and low resolutions to create increasingly eye-catching visuals. These days, it's still popular in games and as an artform in and of itself, despite the possibility of realistic 3d graphics. Why? Well, nostalgia aside, it

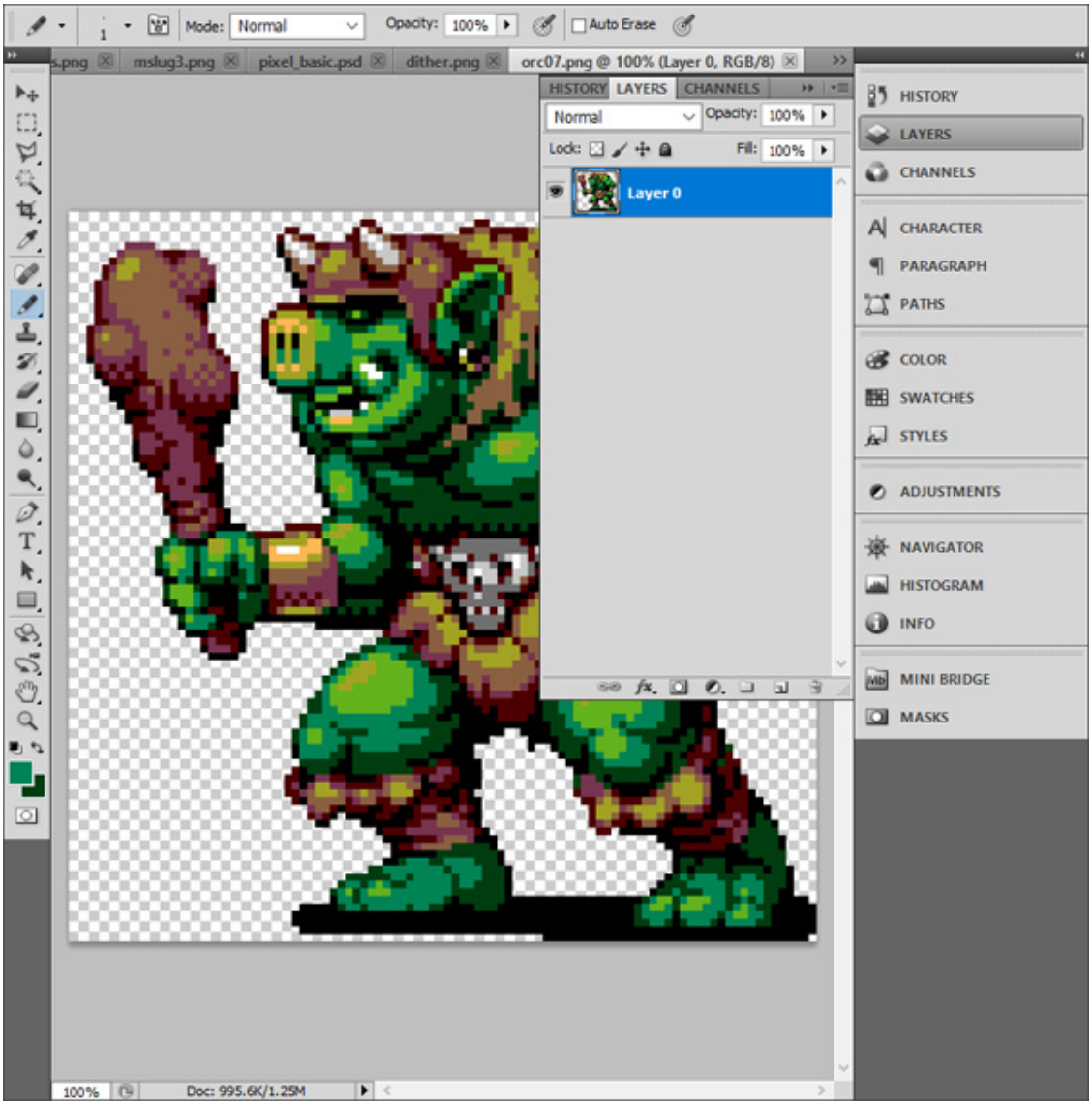
remains a fun and rewarding challenge to create vibrant artwork within such tight constraints. In the same way we admire how a few brushstrokes from a trained hand can represent a form and evoke emotion, so do we admire how a few pixels can combine do to the same.

The barrier to entry for pixel art is also relatively low compared to painted or 3d graphics, making it a nice option for indie game developers seeking to bring their ideas to life. But make no mistake, that in no way implies that it's easy to actually FINISH a game with it. I have seen many an indie Kickstart their pixel art Metroidvania thinking they have a year to finish when in reality it's more like six years. Pixel art at the level most people want to do it is time-consuming and there are very few shortcuts to making it. At least with a 3d model you can rotate it, deform it, move its limbs around, copy animations from one model to another, etc. High level pixel art almost always requires a lot of painstaking pixel placement on every frame.

With that warning out of the way, a little bit about my style: I primarily use pixel art for making video games, and it's from video

games that I draw most of my inspiration. In particular, I'm a fan of the Famicom/NES, 16-bit consoles, and 90s arcade games. My favorite games of that era had pixel art that I would describe as colorful, bold, and clean... but not so clean that it was stiff or minimalistic. That's the style that I modeled my own after, but you could easily apply the ideas and techniques in this tutorial to something completely different. Study a variety of artists and make pixel art what you want it to be!

SOFTWARE



The basic tools required for pixel art are zoom and the Pencil for pixel placement. Also helpful are line/shape tools, select/move tools, and a Paint Bucket for quick fills. There are many free and paid software options you can use that have these tools. I'll outline some of the most popular ones here (including what I use).

PAINT 

If you're on Windows, its built-in paint program is bare bones but has all of the above tools you'd need to make pixel art.

PISKEL 

A surprisingly robust pixel art editor that runs in your browser! Can export to PNG or animated GIF, as well as saving locally to your browser. This seems like a great starting option.

GRAPHICSGALE 

GraphicsGale is the first standalone editor I remember hearing about that was designed just for pixel art and featured animation tools. Created by a Japanese company called HUMANBALANCE, it became freeware in 2017 and is still widely used despite Aseprite's growing popularity. Unfortunately, it's Windows only.

ASEPRITE 

This seems to be the most popular editor available right now. Packed full of features, actively developed, and available for Windows, Mac, and Linux. On top of that, it's open source and can be used for free if compiled from the source code. If you're serious

about making pixel art and don't already have an editor you're attached to, this is probably the way to go.

GAMEMAKER STUDIO 2 💵💵+

GameMaker Studio 2 is an excellent 2d-focused game-making tool that includes a decent Sprite Editor. If you're interested in making pixel art for your own games, it's very convenient to do it all in the same software. I am currently (in 2019) using it to make UFO 50, a collection of 50 retro games. I mostly use GameMaker's Sprite Editor for sprites and create my tilesets in Photoshop.

PHOTOSHOP 💵💵💵+

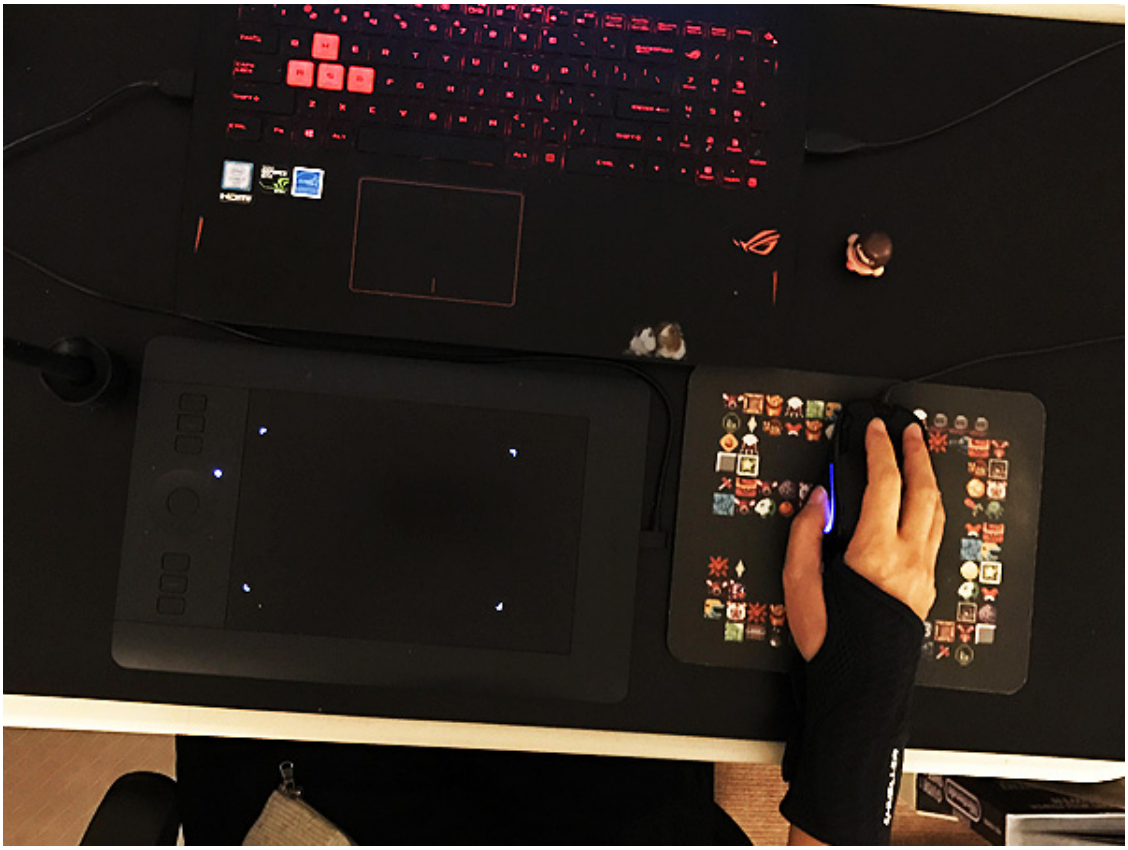
Since Photoshop is an expensive, subscription-based app that's not designed around pixel art, I don't recommend it unless you already have it for painting or image manipulation. It can get the job done for static sprites and pixel illustrations (like the ones I've made for this tutorial), although it's pretty cumbersome compared to focused apps like GraphicsGale or Aseprite..

OTHER SUPPLIES

My pixel art setup. Very black, I'm now noticing...

DRAWING TABLET 💵💵+

I highly recommend a drawing tablet for any kind of digital artwork



in order to prevent repetitive stress injuries to your wrists. RSI is much easier to prevent than to fix. Once they start feeling sore, you're already headed downhill (my days of drawing with a mouse have made it hard to play any games that require mashing buttons). So start taking care of yourself early - it'll be worth it! I'm currently using a small Wacom Intuos Pro.

WRIST GUARD

If getting a tablet is not possible, at the very least get a wrist guard. My favorite is the Mueller Green Fitted Wrist Brace. I've found other brands to either be uncomfortably tight or not supportive enough for me. You can order wrist guards easily online.

THE 96 X 96 SPRITE

Final Fight (Arcade). Capcom, 1989. (Source)

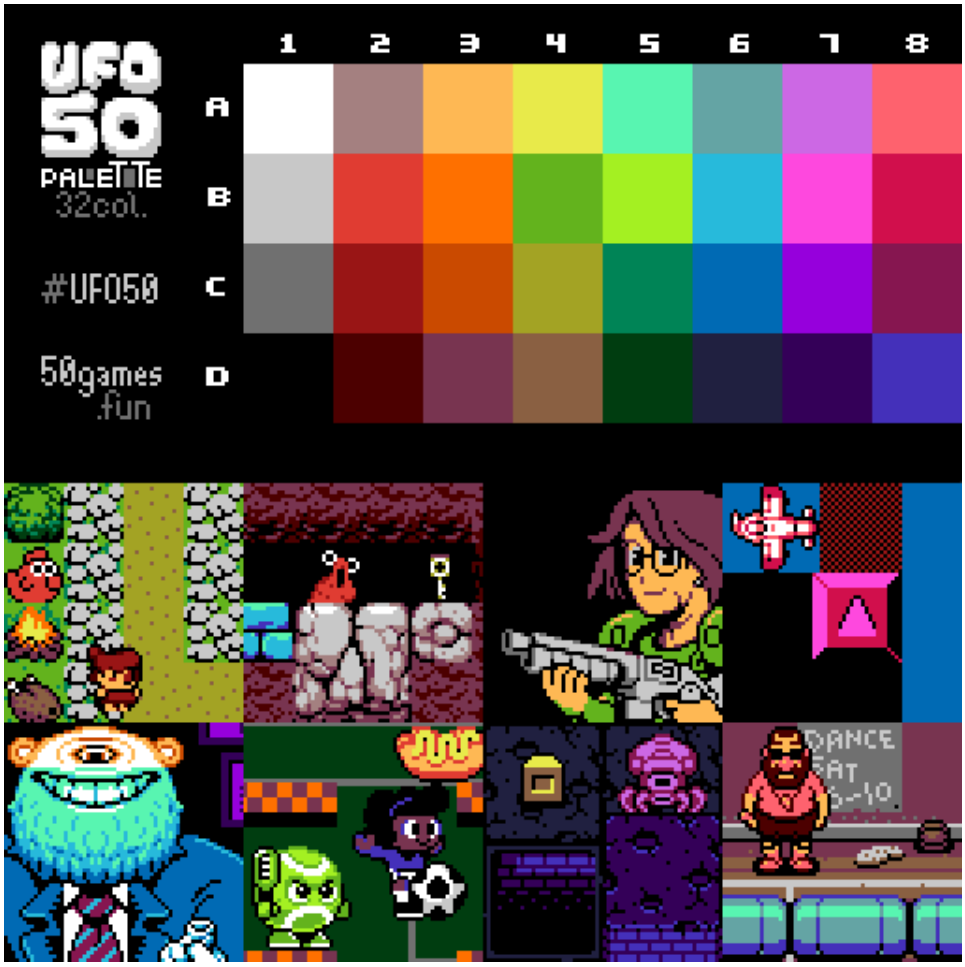
Let's begin! For this tutorial, we're going to start by creating a 96x96 pixel character sprite. I've chosen to make an orc, but feel free to pick something else! I've placed my finished orc in a screenshot of Final Fight above to give you a sense of scale - this is a large sprite for most retro games (the screenshot is 384x224).

The reason we're starting with such a large sprite is that I find it easier to show off the techniques we're learning. Pixelling larger sprites also feels more analogous to traditional artforms like drawing or painting, which might be more familiar to you. After we get the basic tools under our belt we can start working smaller.

1. CHOOSING A PALETTE

Pixel art is defined by its constraints. A pixel has much more meaning in pixel art than other digital mediums and the same is true for colors, which you ultimately want to constrain, as well.

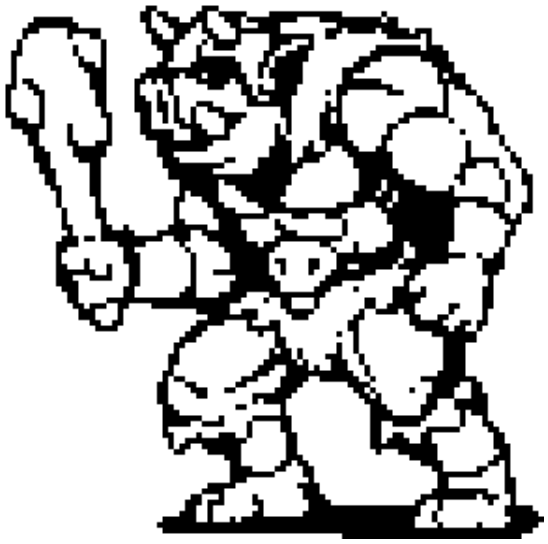
So yes, a color palette is



important and helps define your style. BUT, for beginning pixel artists, I think it's best to put any theorizing about palettes aside and just pick an existing one (or even a few colors at random) so that you can start pixelling. One nice thing about pixel art is that it's very easy to swap palettes at any point, so there's no need to let this decision paralyze you before you start putting some dots down.

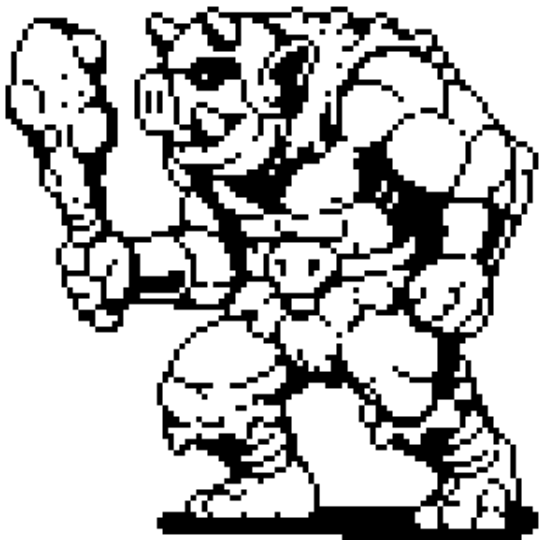
For this tutorial, I'll be using the 32-color palette we created for UFO 50. 32 colors is a popular choice for pixel art palettes, but 16 colors is also common. This particular palette was designed for a fictional console that would have lied somewhere between a Famicom and a PC Engine. You're welcome to use it freely to quickly bypass this step! (Or not! This tutorial is not dependent on the palette at all.)

2. A CRUDE OUTLINE



We'll start our sprite by dragging the Pencil tool around, drawing a sketch the same way we'd draw one with pen and paper. There is definitely an overlap between pixel art and traditional art, especially with larger sprites like this one. From what I've noticed, strong pixel artists are at least pretty good at drawing and vice versa. So it never hurts to improve your drawing skills, as well.

3. CLEANING UP THE OUTLINE



Next, we're going to clean up the outline by removing stray pixels and reducing every line to a single pixel in thickness. But which pixels do we remove, exactly? To answer that, we need to learn about pixel lines and "jaggies".

JAGGIES

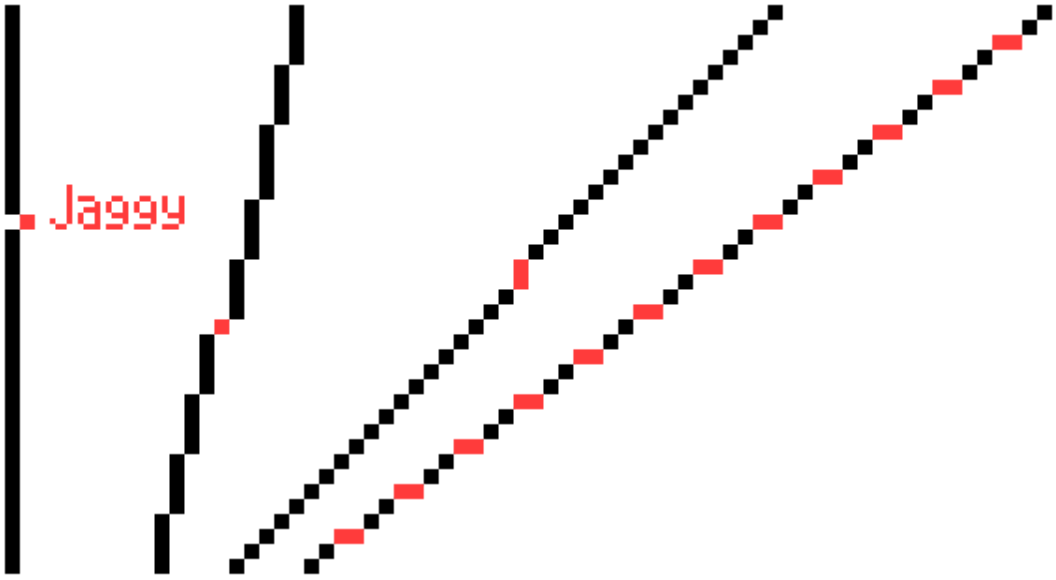
There are two basic lines that we need to learn how to make in pixel art: straight and curved. With a pen and paper, this is

mostly an issue of muscle control, but we're working with little blocks of color, which creates a new kind of challenge.

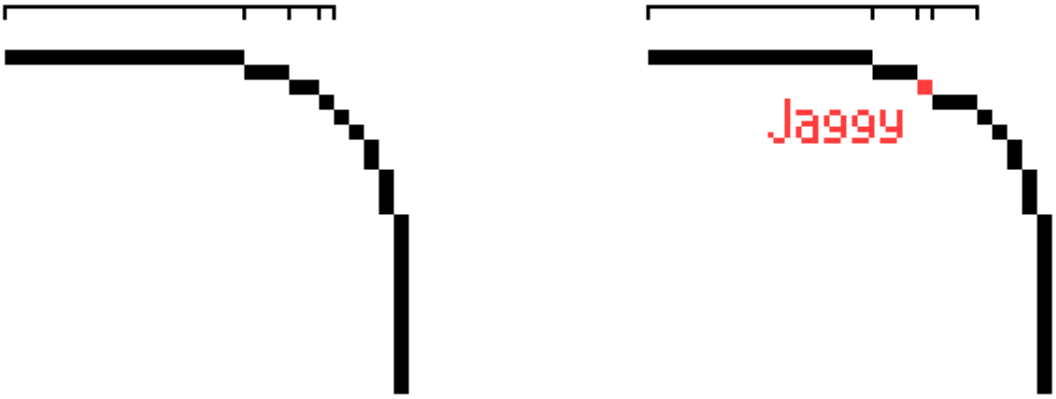
The key to making nice pixel lines is reducing the number of **jaggies**: single pixels or small segments of pixels that break up the consistency of a line. Since a single pixel in pixel art has a great impact on the overall image, jaggies can be an eyesore. Imagine drawing a straight line on a piece of paper when all of a sudden someone slams the table - that little uncontrolled squiggle is sort of what a jaggy in pixel art can feel like.

Let's look at some examples:

STRAIGHT LINES



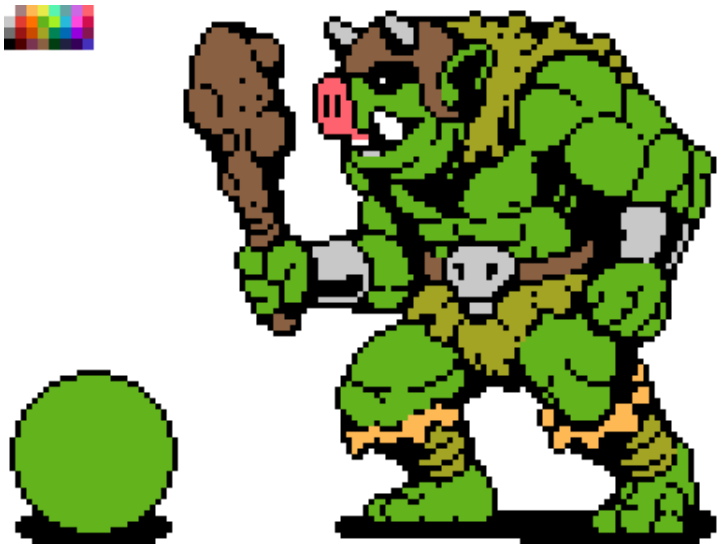
CURVED LINES



With curved lines, jaggies crop up when the length of the line segments don't grow or shrink in a consistent manner.

At this point, you probably think jaggies are worse than stepping in gum, but in actuality, it's impossible to avoid them entirely unless your pixel art is made only of the simplest shapes. Any of your favorite retro games will have jaggies. The goal is simply to minimize them while expressing what you want to express.

4. APPLYING THE FIRST COLORS



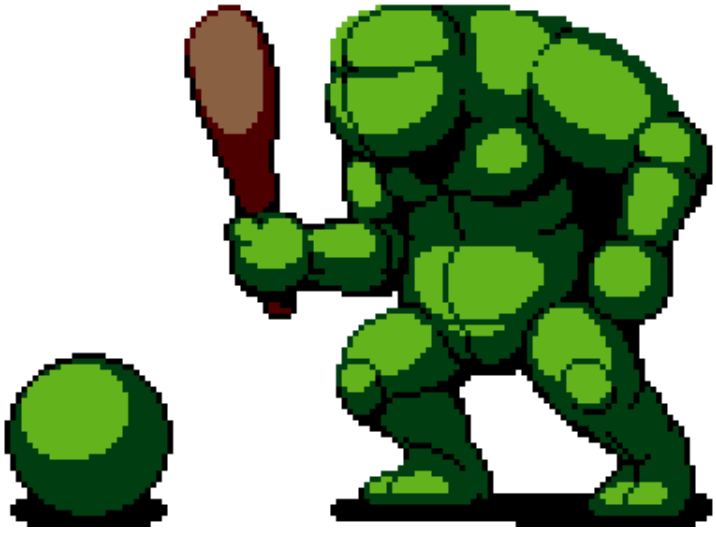
With your Paint Bucket or some other fill tool, color your character in! A palette will make this part simpler, and if your paint software doesn't support palettes, you can always paste your palette into the image itself (as I've done here) and select colors using the Eye Dropper tool.

In the lower-left, I've also introduced a familiar friend, The Ball, to give a quick look at what's going on in each step.

5. SHADING

It's time to shade! The basic idea is that we're going to add darker colors (called **shades**) to the sprite to simulate shadow, thereby making the sprite look 3d instead of flat. For this tutorial, let's assume that there is a single light source above the orc and slightly in front of it, so everything on top and/or in front is bright. We'll add our darker colors to the bottom and back of the orc.

FORM AND VOLUME

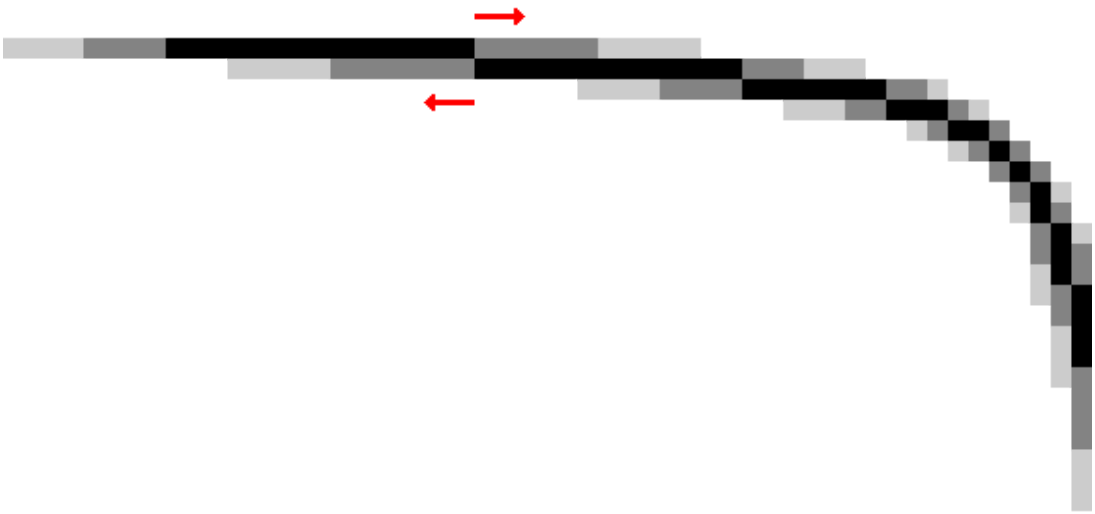


If you're having trouble with this part, you may need to practice thinking about drawings as **forms** with **volume**, instead of simply lines and color. Forms exist in a three-dimensional space and can have volume that fills up that space. By shading, we're bringing out that volume.

It may help you to visualize your character without all of its details and pretend that it's made out of clay instead of pixels. By shading, you're not just adding color - you're sculpting out a form. A well-defined character has details that do not obscure the basic forms - if you squint, a few large clusters of light and dark should still emerge.

ANTI-ALIASING

Every time I introduce a new shade of color, I do some **anti-aliasing** (also known as **AA**), which is a way to smooth out blocky pixels by putting "in-between" colors at the corners where two line segments meet:



The gray pixels soften the "breaks" in our line. The longer the line segment, the longer the segment of AA we use to soften it.

To the right you can see what AA looks like as applied to our orc's upper arm. I use it to smooth out the lines that define the curvature of his muscles.

Be careful not to anti-alias the outside of a sprite used for a game or anywhere you don't know what color the background

is going to be. If, for example, you anti-alias on a light background then that anti-aliasing will stand out on a dark background.

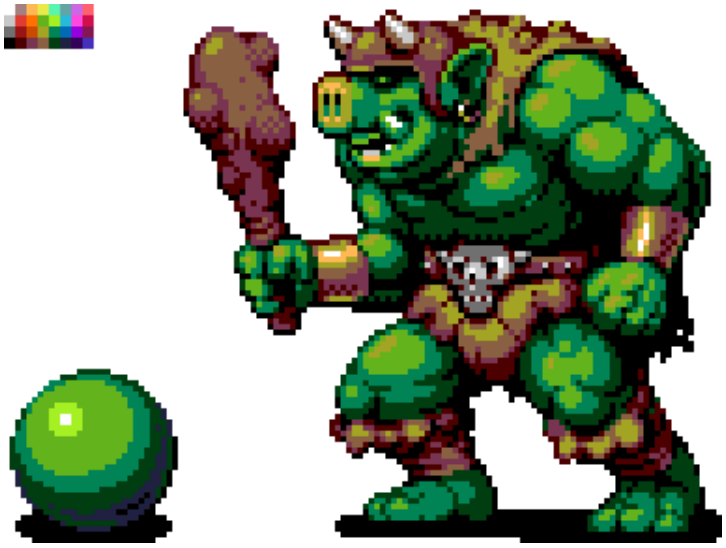
6. SELECTIVE OUTLINING

Up until now, our outline has been pure black, which gives the sprite an overall cartoony look. It's also creating a lot of harsh segmentation. For example, the black lines on the arm are defining the musculature in an extreme way, making them look less like they are all part of the same body part.

To give the sprite a more naturalistic look and to soften the segmentation (in order to bring out our character's basic form), we can use a technique called **selective outlining** or **selout**. Selout means replacing a lot of the black outline with lighter colors. Toward the top, where the light is hitting our sprite, we will use the lightest colors or, where the sprite meets the negative space, we may remove it entirely. For segmentation (e.g. for muscle, fur texture, etc.), we can use our darker shadow colors instead of pure black.

I've also added another level of even darker shadow to the orc in this step. So there are now three shades of green on our orc's skin. This new shade of green can be used for selout and further anti-aliasing.

7. FINAL TOUCHES

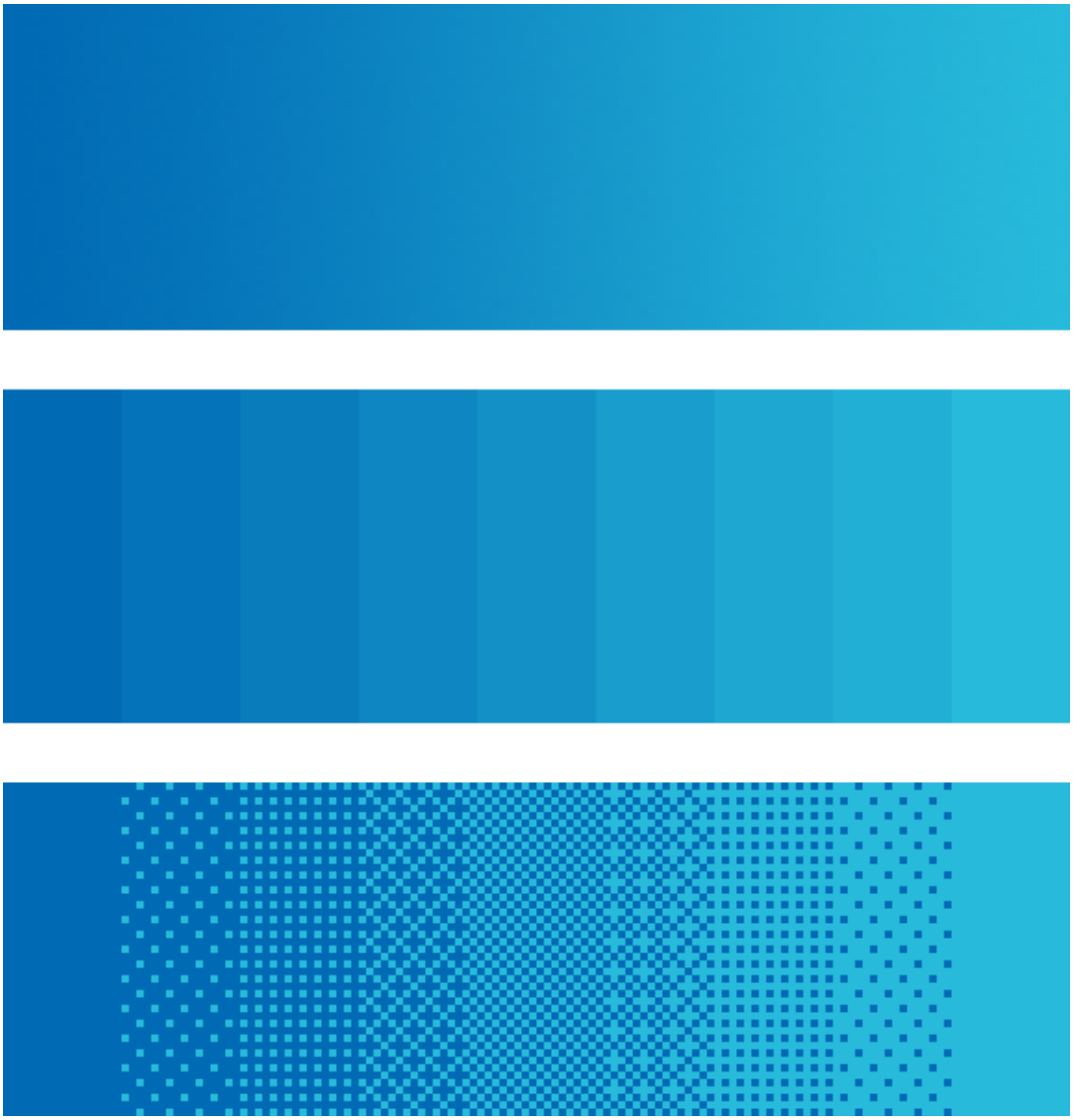


At the end, we can add highlights (the brightest spots on our sprite), details (earrings, studs, scars), and continue to make adjustments until we're happy with it (or need to move on, as is often the case!).

A couple other things to try at this stage: flipping your artwork horizontally is a powerful trick in digital artwork that often exposes flaws in proportions and shading. Another trick is to remove the color from your artwork (i.e. set the **saturation** to zero) to see whether your shading still reads well.

DITHERING

Until now, we've mostly been shading with large, unbroken clusters of darker color. However, there's another technique, called **dithering**, that allows us to bridge two different shades of color without adding a new shade. Take a look at the following example:



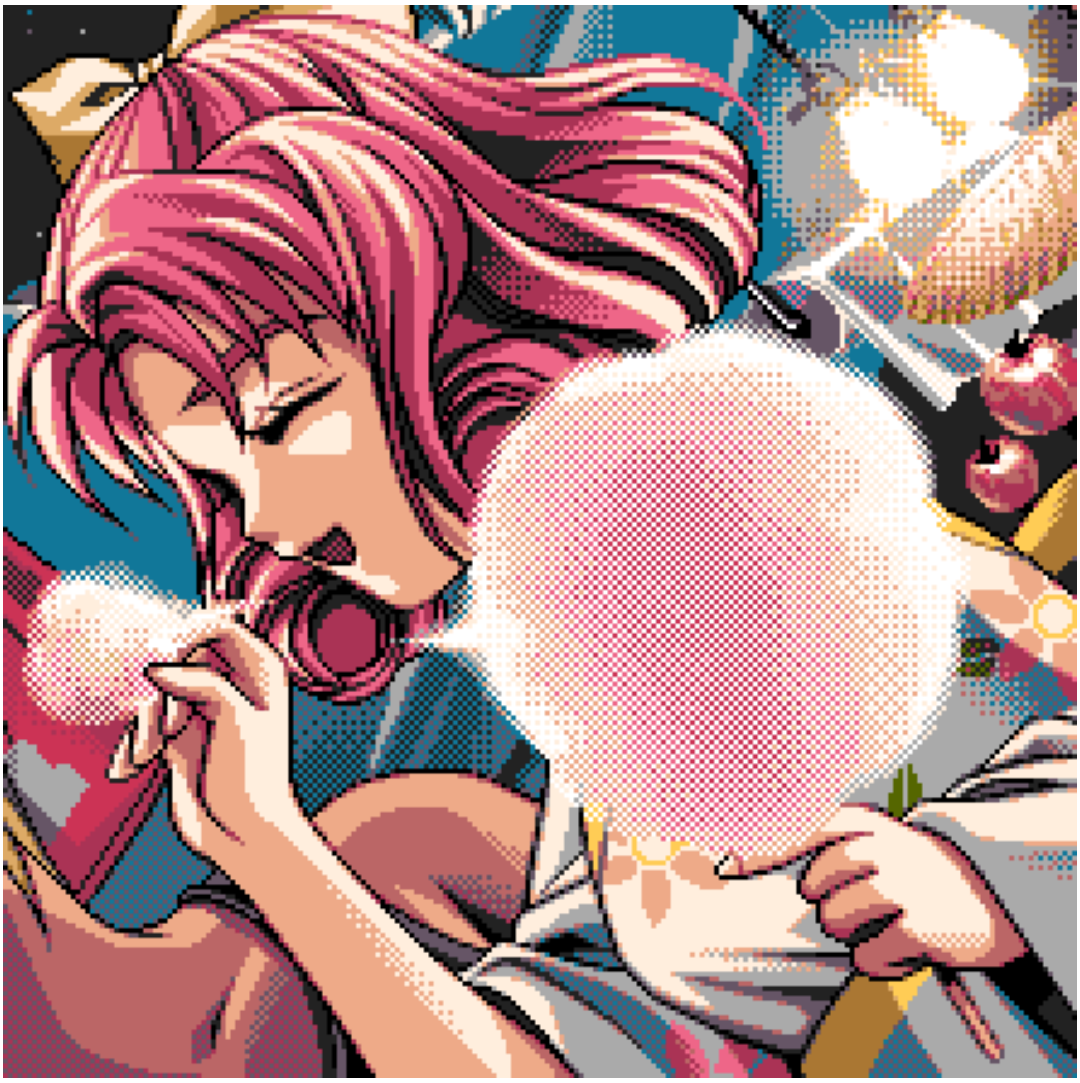
At the top is a **gradient** that moves from dark to light, using hundreds of different shades of blue. In the middle, we've reduced the number of colors to 9, but that's still a lot of shades for a single color. It's also created a distracting effect called **banding**, where, because of the thick, uniform bands of color, our eyes begin to focus on the lines where the colors meet instead of the colors themselves.

Finally, at the bottom we've applied dithering, which mitigates the banding effect and only uses 2 colors! The idea is to create **noise** of varying densities to simulate the gradation of color. It's very similar to a technique called "halftone" that's used in printing. Or "stippling" in illustration and comics.

I use dithering sparingly - on the orc I added only a little bit for texture. Some pixel artists don't use dithering at all. Some use

it extensively and make it look quite good. In general, I think it works best on large areas of a single color (take a close look at the sky in the Metal Slug 3 screenshot from above) or in places that we want to look rough or bumpy (like dirt, perhaps). If you like how it looks, experiment with it and find out how to make it work best for you!

If you want to see dithering used extensively and done well, study the games of the Bitmap Brothers, a UK game studio from the 80s, or the games on the PC-98, a Japanese computer (please note that many PC-98 titles are NSFW):



Kakyusei (PC-98). Elf, 1996. (Source) There are only 16 colors in this image!

8. ONE LAST LOOK

One of the dangers of pixel art is that, due to its constrained, grid-like nature, it is easy to feel like you can get it "just right", and you may find yourself spending an excessive amount of time tweaking your sprites at the end. In some ways it feels like a puzzle to solve, and that can be very addictive. As a result, pixel art tends to attract perfectionists, so please be careful about lingering on a single sprite for too long. In game development, a single static sprite is just one small piece of a very complex arrangement of pieces and it's important not to lose sight of the bigger picture, so to speak.

Even if you aren't making pixel art for games, it's good to be able to say "This is good enough!" and move on. The best way to

improve your skills is to see the entire process from beginning to end as many times as possible, on as many different subjects as possible. At the very least, leaving a piece behind for awhile will let you look at it with fresh eyes!

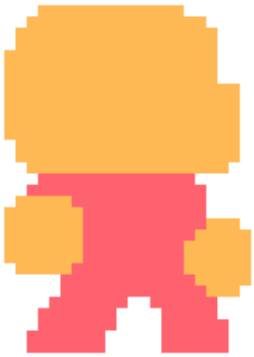
THE 32 X 32 SPRITE

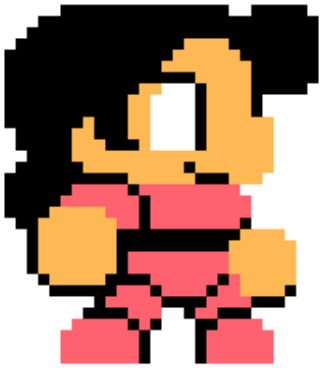
We created a large 96x96 sprite first because at that size it still feels like drawing and painting, but with pixels. The smaller a sprite gets, the less your sprite looks like what it's supposed to represent and the more responsibility each individual pixel has.

In Super Mario Bros., Mario's eye is just two pixels stacked on top of each other. So's his ear. And his creator, Shigeru Miyamoto, explained that the reason why he has a mustache is because they needed it to distinguish his nose from the rest of his face. So one of Mario's most iconic features was not just a character design choice but also a pragmatic one! Proving the old adage that necessity is the mother of invention... and giving us further insight into why pixel art is so interesting.

With all that in mind, the basic steps we'll take to create a 32x32 sprite are actually pretty similar to the 96x96 sprite: sketch, color, shade, and then polish. For the intitial sketch however, I often use colored shapes instead of drawing the outline because at this size, color plays a larger role in defining the character than outlines. If we look at Mario again, he doesn't have an outline at all! And it's not just his mustache that's putting in work - his sideburns define his ears, his sleeves define his hands, and his overalls more or less make his entire body intelligible.

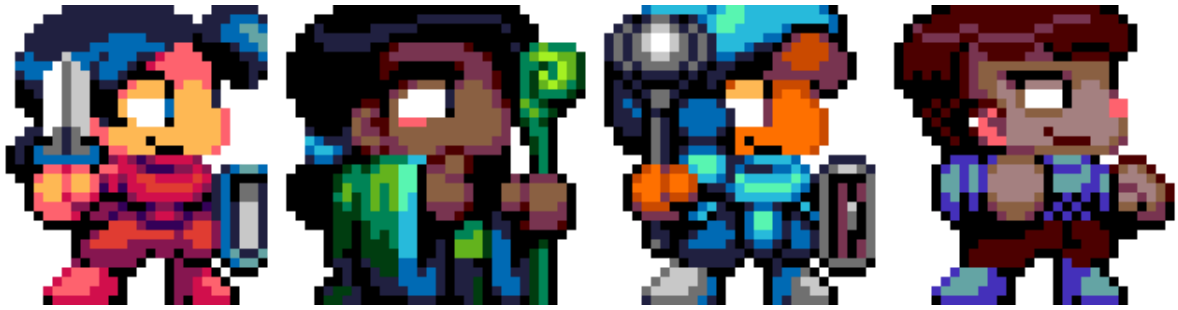
Making small sprites is about making compromises. If you add an outline around something, you may lose the room to shade it. If your character has well defined arms and legs, the head will probably need to be smaller to make room for them. By using color, selout, and anti-aliasing effectively, however, you can make your canvas feel larger than it actually is.





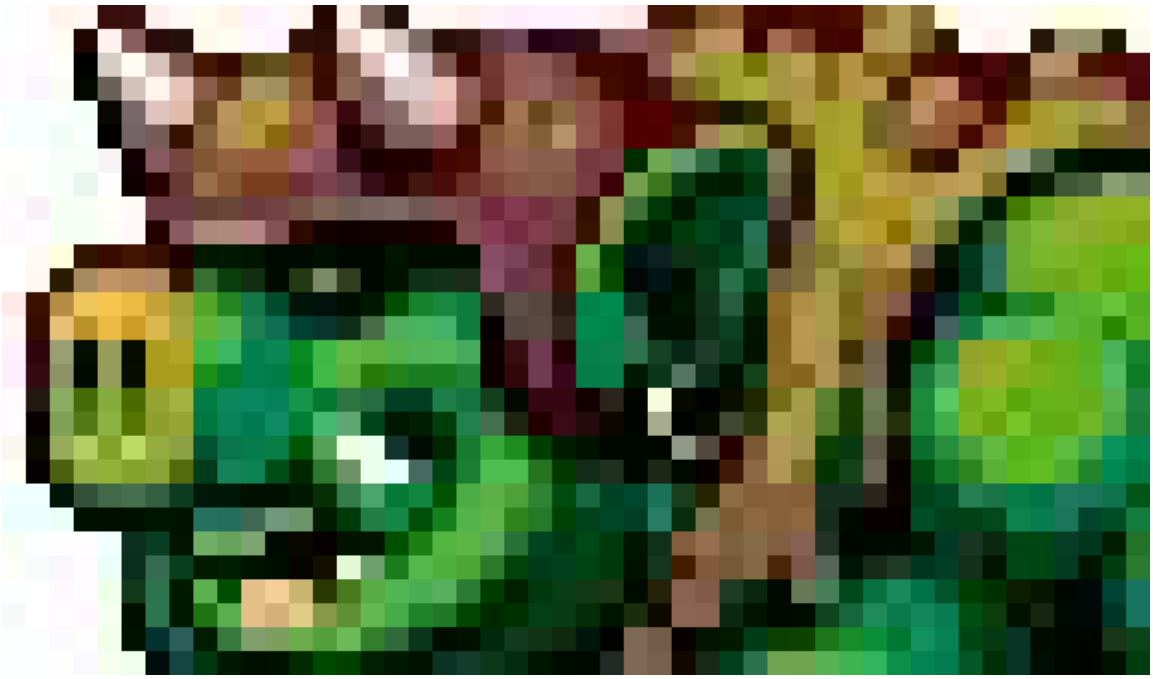
For small sprites, I tend to favor **chibi** (or **super-deformed**) designs where the characters are cute and have large heads and eyes. It feels like a great way to create expressive characters in a limited space (it's also an appealing art style regardless). But perhaps you're more interested in bringing out a character's

mobility or brute strength instead, in which case you may choose to focus less on the head in favor of a more powerful-looking body. Ultimately, it's up to your preferences and your project!



The full party assembled!

FILE FORMATS



Enough to send chills up any pixel artist's spine.

The above is what will happen if you save your artwork as a **JPG**, a **lossy** file format. What that means is that data is actually lost when you save it, due to the way the file is **compressed** (to

reduce file size). Practically speaking, your nice, crisp pixel art will end up looking blurry and you won't be able to get its original palette back easily.

The recommended **lossless** file format for static pixel art is PNG. For animations, animated GIFs are the most popular format.

SHARING YOUR PIXEL ART

Sharing your pixel art on social media is a great way to get feedback and meet other pixel artists (don't forget to use the **#pixelart** hashtag!). Unfortunately, social media websites tend to convert PNGs to JPGs without asking, tarnishing your artwork as it goes public. On top of that, it can be hard to figure out what it was about your image that triggered the conversion!

To help out, here are some tips on how to keep your pixel art crisp on various social media sites. Note that these sites change their algorithms frequently and this section may not always be completely up to date.

TWITTER

The key to keeping PNGs intact on Twitter is to make sure they're either less than 256 colors or less than 900px on the longest side. (Source) I would also upscale your images to at least 512x512 px, making sure that you are upscaling at a clean multiple (200% and

not 250%, for example) and preserving hard edges (called "Nearest Neighbor" in Photoshop).

Animated GIFs must be less than 15 MB to post on Twitter. As for quality, the prevailing theory is that they should be at least 800x800 px and looping animations should be looped three times, with the final frame of the GIF displayed at half the length of every other frame. However, it's unclear how necessary all these steps are as Twitter continues to update how they display images. At the very least, I would make sure the animations are at minimum size. (Source)

INSTAGRAM

As far as I can tell, there is no way to post lossless images to Instagram, but you can improve the look by scaling up your artwork to at least 512 x 512 pixels.

That's the end of this tutorial!

Click on the hand to check out the next tutorial: **PIXEL ART: COMMON MISTAKES**

Return to Top

Last updated: Jan 28th, 2020.

PIXEL ART: COMMON MISTAKES

We've all been there: you learn a few things and it all seems so simple, but when it comes time to make something yourself, it doesn't look right! In this tutorial, we'll start with some "beginner's pixel art" and we'll make edits to it in order to show off some common mistakes and their solutions. One of the hardest parts of making art is self-critique - we're often too close to our art to look at it objectively. Asking other artists to edit your work by way of feedback can be a great way to illuminate problems that you may not see. (But keep in mind that it's considered rude to create an edit of someone's work unless they specifically asked for one.)

This is a follow-up tutorial to PIXEL ART TUTORIAL: BASICS.

REWORKING NAIVE PIXEL ART

What I've done here is create a little scene that looks like a lot of early attempts at pixel art. Don't get me wrong, this type of work can be very charming, which is why I use the term "naive". But it also looks flat and stiff and it lacks readability. Furthermore, part of what I enjoy about art is admiring **craftsmanship** - the skill and knowledge of the artist at work. Naive art can evoke strong emotions, but I'll always miss the feeling of awe as I marvel at someone's technical mastery.

Ultimately, my favorite artworks reveal a unique and genuine vision but also display strong craftsmanship. The former requires a lot of personal exploration and is hard to teach in a tutorial, so we'll focus on the latter! What I'll do here is rework this scene over a few big steps and along the way we'll try to understand what makes this work naive.

1. SIMPLIFYING AND RECOLORING

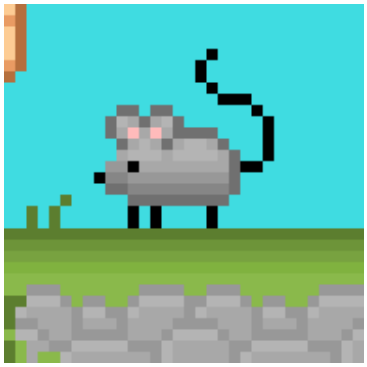
To begin with, I won't change any of the outlines. Instead, I'll remove the shading and, at the same time, reduce the color count and increase the contrast of those colors. You'll notice this immediately improves the readability of the various objects in the scene.

THE PROBLEM: TOO MANY SIMILAR COLORS

We want each color to have its own identity so that it can do as much work as possible. The efficiency of each dot is what makes pixel art unique as an art form and why we continue to work with limited palettes and limited resolutions even though it's not necessary in the modern day. When colors are too similar, pixels begin to blend together and get lost. Sometimes this may be an effect you want - for example, it might make sense

for a background image to look more indistinct so that your characters stand out better against it. But we want to make sure that it's something we're doing intentionally.

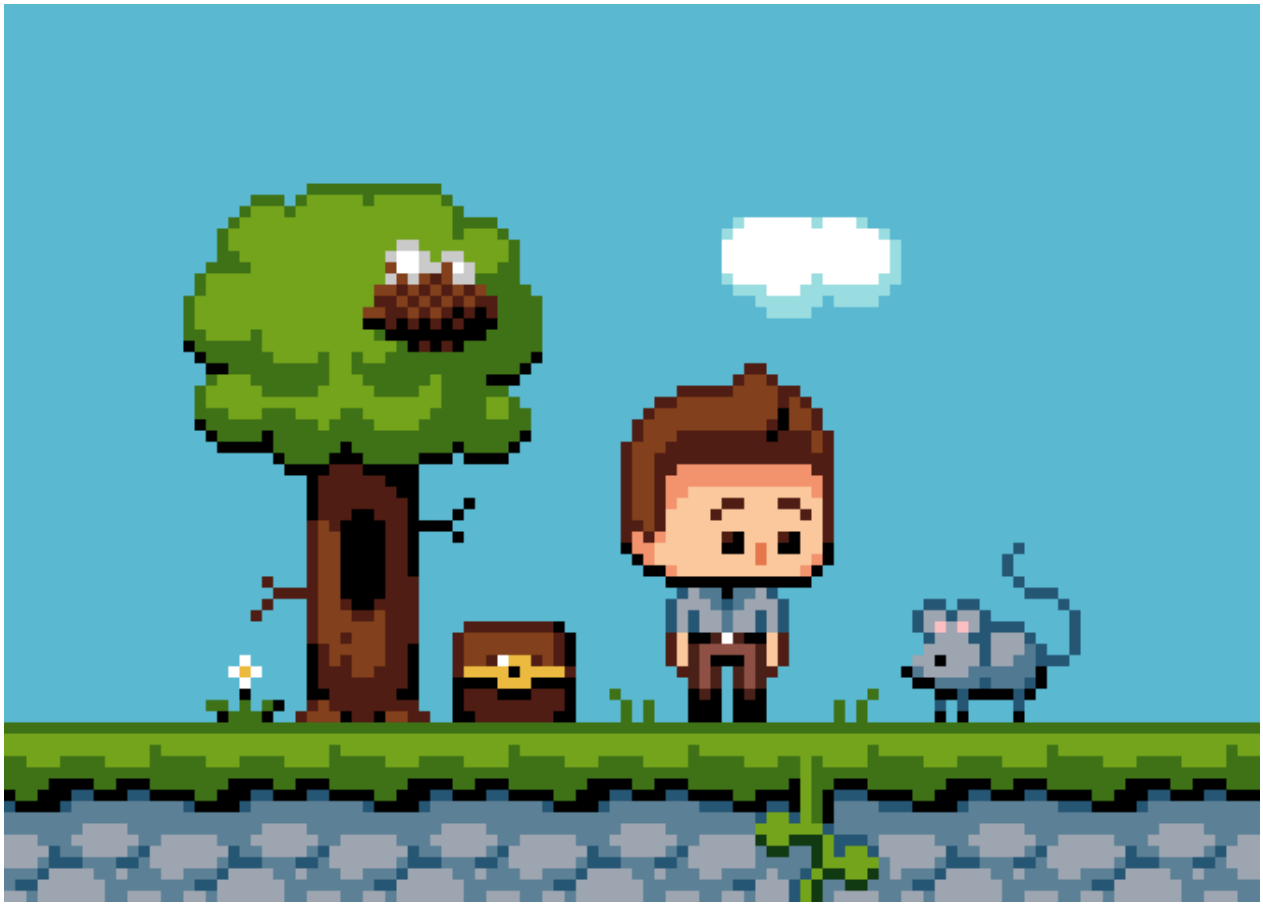
THE PROBLEM: NAIVE COLORING



When we start out drawing, we tend to think in terms of what color something "should be": leaves are bright green, the sky is bright blue, rocks and mice are pure gray, etc. In reality, color is rarely pure and reflected light can cause colors from nearby objects to mix with one another, adding complexity.

This doesn't mean that one needs to study hundreds of hours of color theory to make professional-grade pixel art. You could, for example, use or modify existing palettes. Or simply be observant and do some trial-and-error color picking, testing out varying levels of brightness and saturation until you arrive at a pleasing combination. In short, experiment!

2. CREATING FORMS WITH VOLUME



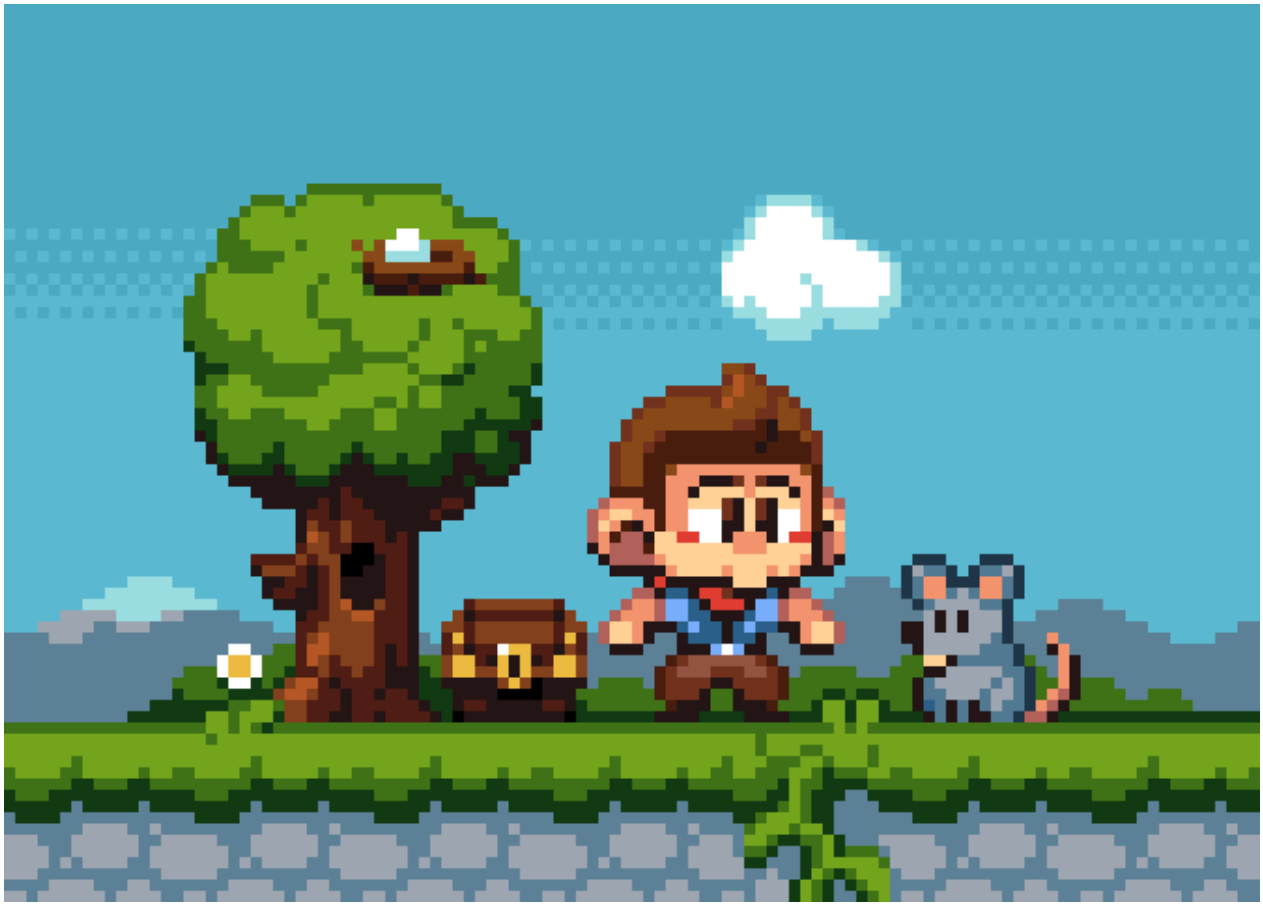
In PIXEL ART TUTORIAL: BASICS, we talked about thinking in terms of forms that have volume (i.e. they take up 3D space). The best way to express volume is to add realistic shading and most objects can be reduced to a simpler shape when thinking about shading so that the basic form does not get lost in too many details. For example, a treetop could be thought of as a few green spheres glued together instead of as thousands of tiny leaves. Or a bird's nest could be thought of as a brown bowl instead of a bunch of small twigs.

THE PROBLEM: PILLOW SHADING

Pillow shading refers to shading from the outline inward, creating a "pillowy" effect. This type of shading almost never occurs naturally in real life and tends to make the object look blurry and indistinct. It's often paired with the

"too many similar colors" problem.

3. DYNAMIC CHARACTER DESIGN



We can do a lot with color and shading, but stiff and flat designs will prevent us from doing more.

THE PROBLEM: CARDBOARD DESIGNS

With pixel art, we're making artwork on a grid, and it's easy to let that grid force us to design along straight lines. Again, thinking less in terms of lines and shapes and more in terms of forms with volume can be very helpful in that regard. Also, try to bring out the

personality of each object - imagine them moving around and exaggerate the features that best represent them. The more "alive" your artwork feels in your head, the less constrained you'll feel by the medium.

One loose rule I try to follow is to avoid rendering anything with only a single pixel of thickness - I call this my "Chunky Pixels Rule"! New pixel artists tend to make protrusions like arms and legs (and tree branches) very thin. This makes them hard to shade and consequently hard to sculpt into three-dimensional forms. As a result, they feel flat and flimsy.

4. BEFORE AND AFTER

Using the slider, you can compare the Before and After images. What do you think? Is the final edit an improvement? It's okay if there are things you genuinely like better about the original - my

ultimate goal isn't to get you to make pixel art like me but to get you to start thinking about what looks good to you and why. Every time you look at pixel art, whether it's your own or other people's, ask yourself what stands out about it, good and bad. What could be improved upon? What seems "wrong" but adds something to the work regardless?

Self-evaluation is one of the many hard aspects of being creative. Try too think about it as an ongoing personal journey instead of a competition. There's no end point, just the enjoyment of making art and learning!

That's the end of this tutorial!

Click on the hand to check out my **PIXEL ART GALLERY** and study some master-level pixel art from the 90s and early 2000s.

Return to Top

Last updated: Jan 28th, 2020.