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

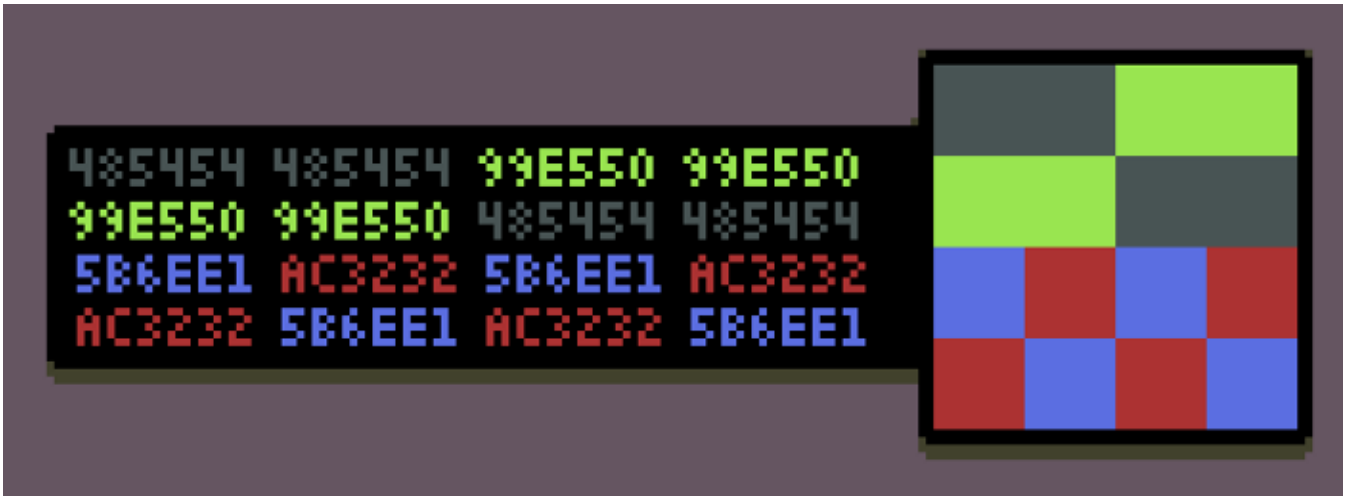
Let's start talking about how images are saved in the computer. If we simplify a lot, there are basically two types of 2D image files: bitmap and vector.

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.



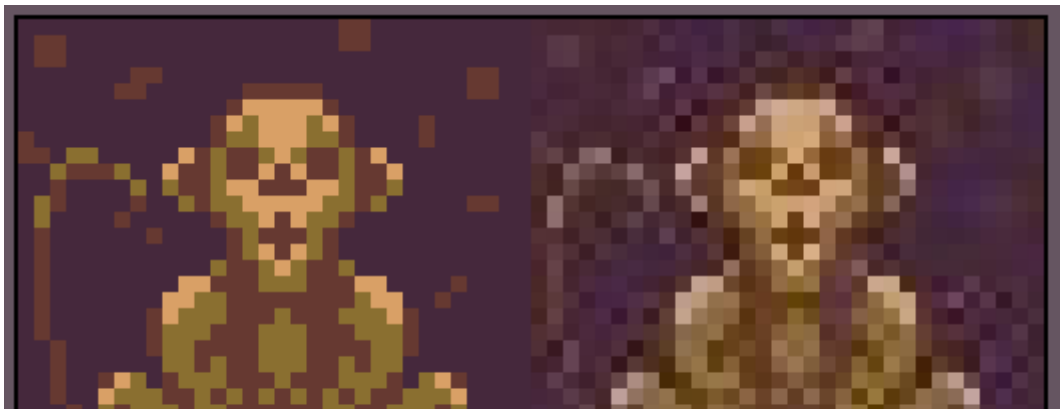
A simplified example of how a bitmap is saved

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

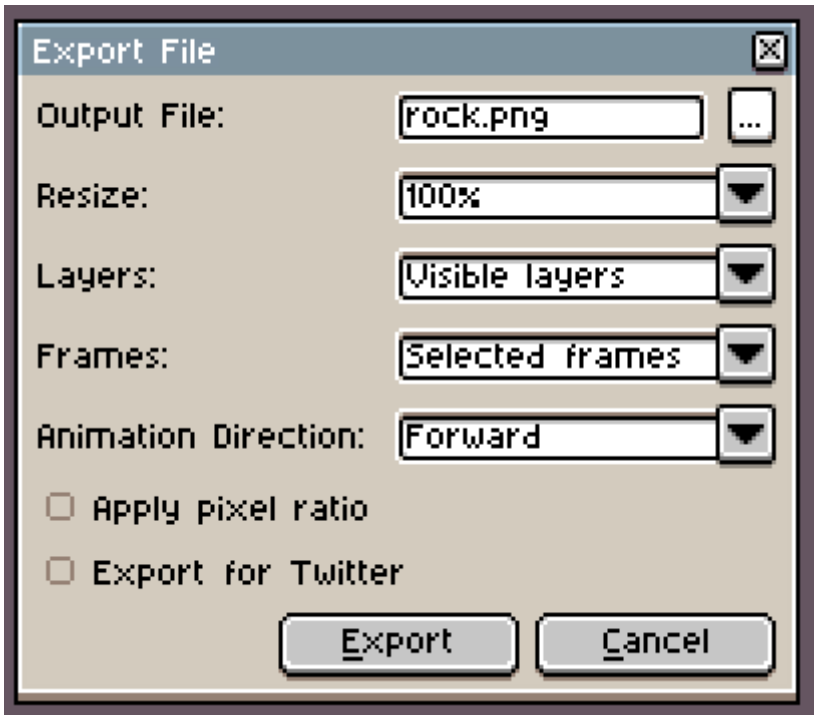
Pretty intuitive, right? Use this to save your files in **.aseprite** format and keep all the file information, so you can can come back to it later if you want to make changes.

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!

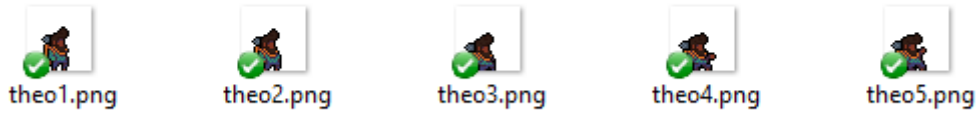
Saving for the engine

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.

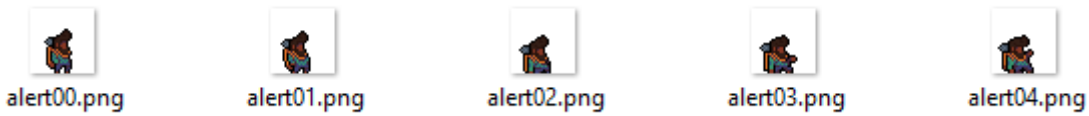
Saving an image sequence

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:

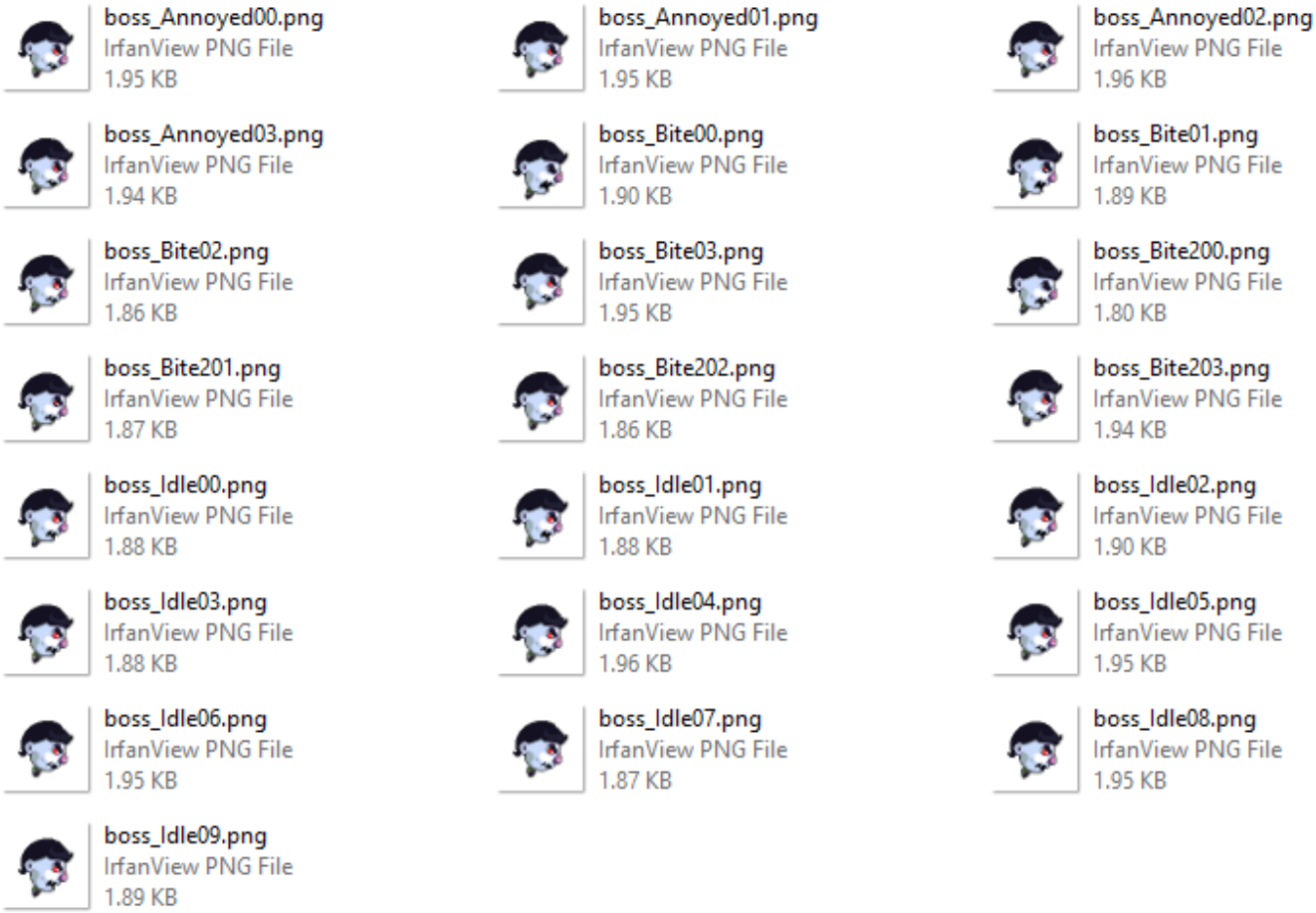


Feel free to experiment by adding different numbers in the end of the file.

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

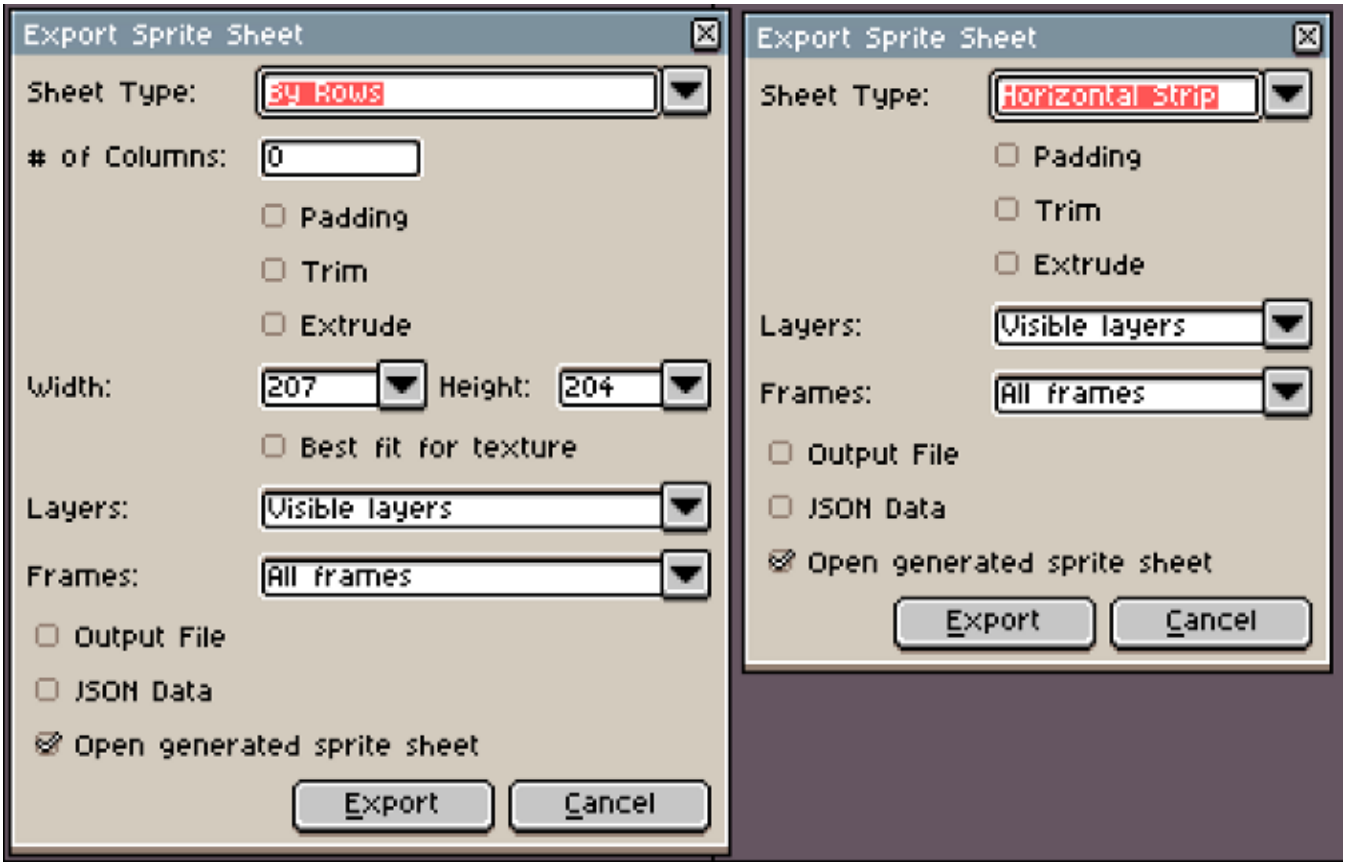
That's great for organizing, and exporting it is also very easy, just type something like **boss_{tag}{tagframe00}.png** and Aseprite will export files like this:



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.

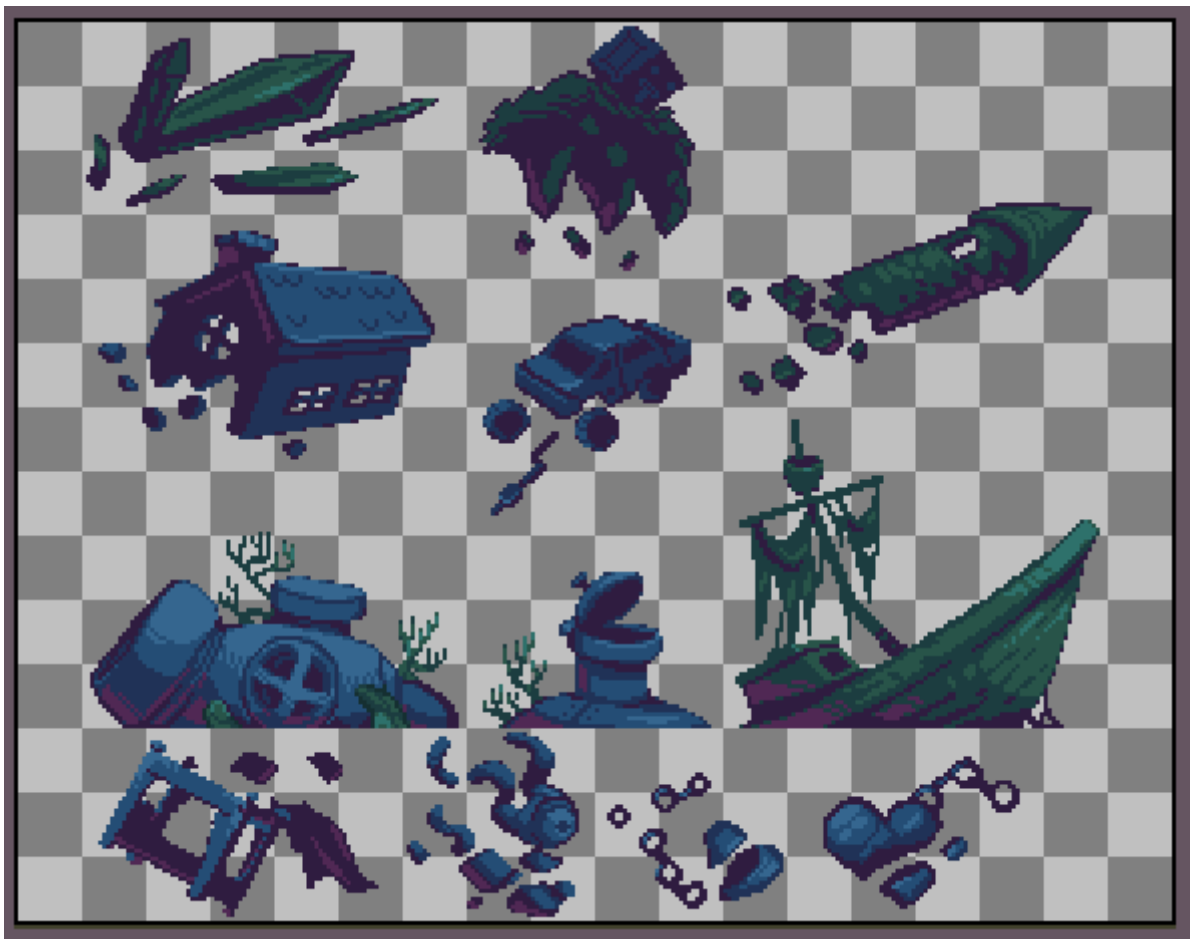
About the number of rows and columns, if your engine doesn't require anything specific I recommend checking "Best fit for texture" and let Aseprite calculate the optimal size.

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.

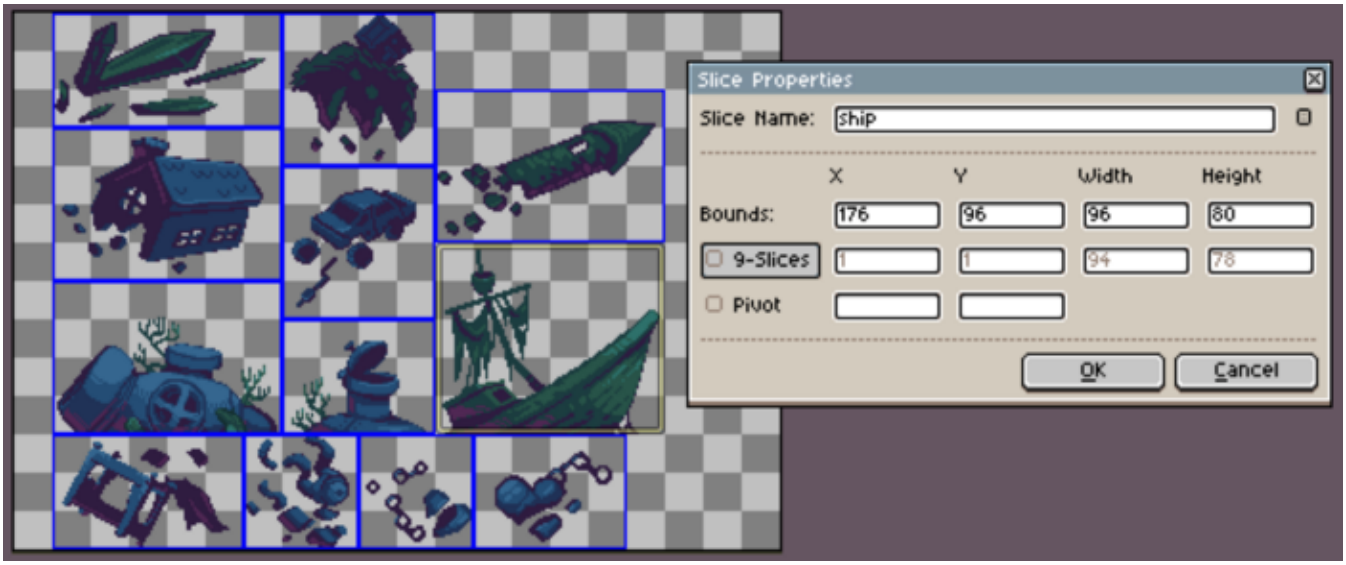
The first step is to make the sprites in a single file, side by side. Just don't let them overlap each other in a rectangular area.



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