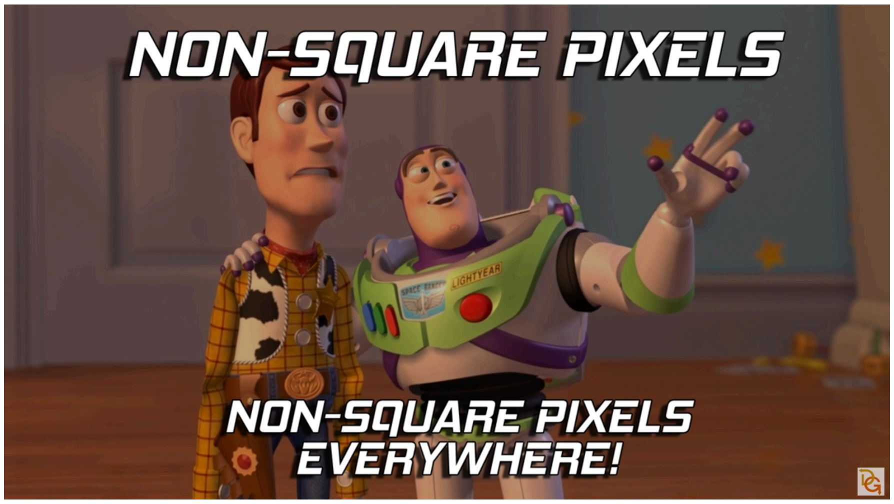
Scaling, Shaders, Filters, Aspect Ratio, and You:

Why I'm right, and you're wrong about Integer Scaling

(Except, not really because what matters is the image that pleases you.)

(auth: Cantankerous Rex)



Other Image Credits and further reading:
Chrono Trigger and Super Metroid AR Comparisons:
https://orcaapp.herokuapp.com/snes
Megaman X Bilinear Filtering sample:
https://tanalin.com/en/articles/integer-scaling/

Other image Thanks - Gabraham, SML
-Other Aspect Ratio discussion-
Displaced Gamers - Explorations of Aspect Ratio from Arcade, SNES, and DOS games. Highly recommended; much of my information comes from these videos and DG deserves recognition for his attention to detail, and deep-dives into everything that is classic gaming.
https://www.youtube.com/watch?v=LHfPA4n0TRo -CPS1 as an example of PAR vs AR
https://www.youtube.com/watch?v=ssluTgfkdlg -Why *neither* 8:7(PAR) or 4:3(AR) is always best
https://www.youtube.com/watch?v=YvckyWxHAIw - Why 640x480 isn't always 'correct' for DOS.

If you spot anything in here that is patently incorrect or could be worded better, just shoot me a PM via discord or Reddit and I'll happily correct it.

Christ this thing is long enough to warrant a table of contents now.

What is this guide? **3**

Preamble and Basic Settings **4**
"Emulation handhelds"
"A Note on the GBA Experience"

System Specific Settings **5**

Settings for JZ4770 Devices **6**

Why these settings, specifically? **7**

The limitations of 640x480 .... **8**

Notes on Integer Scaling **9**
"Integer Scaling at PARs other than 1:1"
"The 'Circle Test' " 10

ENTER: Non-Integer scaling **10**
"Nearest neighbor " 11
"Bilinear/Bicubic/Trilinear Filtering" 11

Finding a Comfortable Medium **12**
"The Difference between Shaders and Filters"
"Shader Recommendation"

Filter/Shader Comparison Screenshots **13**

Boring technical shit about display panel technology **19**
"The RG280's…special display" 20

Changelog **21**

WHAT IS THIS GUIDE?

The original content of this section included information about displays for specific devices, but with the continued proliferation of various inexpensive handheld devices and the rapidity with which they're released, it became apparent that trying to keep tabs on all of them would be stupid. Rather than try to cram information about more devices here, I thought it better to explain…why this guide exists, and why I might waste my time obsessing over picture quality for 30 year old games being run on pocket sized magic linux gameboys.

Handheld emulation devices have been doubtlessly growing in popularity over the last few years - if not explosively since the Dingoo days, but historically they've existed within deep, niche interest spheres. Starting in early 2021, we started seeing coverage by more mainstream internet personalities which doubtlessly generated quite a lot of new interest (Some even having gone viral on TikTok) bringing some much-needed new blood into the communit, and a whole fresh hell of misinformation.

A lot of this misinformation is about image quality and display settings; oft-repeated myths that lurk and are often vehemently defended for reasons that I won't even attempt to fathom. While most are planted with seeds of truth, there is a bit of nuance that is often lost in the repetition which I fear sometimes leads to tinkering decisions that might in fact, harm a user's overall experience or just waste their time.

There are two common myths that I want to tackle:
1 - That Integer scaling is always best
2 - 4:3 Apect ratio, 640x480 resolution displays are the most accurate for classic systems

While this guide delves into much greater detail on why neither of these is necessarily true - or even true in most cases, the very short versions are:
1 - Integer scaling does not reflect a given image's 'natural' display conditions, leading often to a warped image and wasted screen real-estate
2 - Few classic game system's output will scale cleanly to a 640x480 resolution. Not even many classic DOS games did this, even when they were programmed specifically for 480i displays.

Resolution/Aspect Ratio (AR) Index		
Resolution	**Aspect Ratio**	**Relevant devices**
320x240	**4:3**	**RG350p, RG280m/v**
480x272	**16:9**	**Sony PSP**
480x320	**3:2**	**OGA, RG351p/m, RK2020, RGB10(s), RGB20, etc.**
640x480	**4:3**	***Too damn many to list.***
854x480	**16:9**	**OGS, OGU, RGB10Max**
960x544	**16:9**	**PS Vita, RG503, RG505**
1280x720	**16:9**	**Nintendo Switch/lite**
1280x800	**16:10**	**Steam Deck; many handheld PCs**
1334x750	**16:9**	**Retroid Pocket 3 and 4 series**
1920x1080	**16:9**	**Ayn Odin; more expensive handheld PCs**
1920x1152	**5:3**	**RG552**

Base Settings - Getting the ‘best’ image <u>while using Retroarch</u>
I want to stress that this guide and my nitpickiness should *not* be construed as anything like an authoritative “This is the best look for every game ever”. These settings work well for me, but if they’re not to your taste - you’re wrong but I won’t throw you in Mind Jail for it. Before we get into the melee of image enhancement or CRT/LCD recreation, we should start by getting a clean-looking picture that will occupy the full vertical space of a given display, and mitigate scaling artifacts such as pixel wobble/distortion/shimmer, or blurring - without resorting to filters that distort/enhance the original image beyond what is necessary:

Settings
- > Video
  - > Scaling
    - - Integer Scaling: **OFF (See below note on GBA/WS for more info)**
    - - Aspect Ratio: **Core Provided**
    - - Crop Overscan: **ON**
  - > Bilinear Filtering: **OFF**
  - > RGA Scaling: **OFF** *(Option only available on Rockchip based devices)*
  - > Video Filter: **NONE**

Quick Menu
- > Shaders
  - > Load
    - > Shaders
      - > Interpolation
        - - **Bandlimit-Pixel.glslp**
      - > Parameters
        - - Smoothness: **0.2** (or **0.3**, if scaling artifacts are visible)

While most systems will not be negatively impacted (I have tested using Genesis, SNES, PSX, GB/C, GBA, NeoGeo, and CPS(1/2/3) cores) this is a somewhat demanding shader set on lower powered hardware, and there are certain cores that will suffer while using it: SNES cores that make use of MSU-1 features will experience frame drops on RK3326 based devices and the shader can’t be readily used in combination with many filters on anything more demanding than GBC, except with more powerful hardware. This is less of an issue on such hardware, but if your daily driver is an RK3326 or equivalent device, do bear this in mind.

A note on GBA / WS: The GBA and Wonderswan use an unusual 3:2 aspect ratio. Devices using 3:2 AR displays *should* <u>mostly</u> be able to use **integer scaling** with no filters or shaders to achieve a very clean and sharp image that wastes no screen space. 16:9 displays at 480p also work well here with integer scaling - however higher res panels at this AR will require additional tinkering to achieve the same end. Also keep in mind that for >480p 16:9 panels or 4:3 displays, using a non-integer scaled image is likely to interfere with bezels or overlays.

SYSTEM SPECIFIC SETTINGS

GBA/Wonderswan settings for:
OGA, OGS, RGB10MAX, RGB10, RGB20, RG351p/m (NON MP), RK2020

The below settings are for devices using either native 3:2 displays, or 16:9 displays at 480p.

Settings
> Video
>> Scaling
>>> - Integer Scaling: **ON**

A well known ‘quirk’ of GBC and (particularly, earlier) GBA titles, was the use of highly saturated colors to compensate for the desaturating effect of the GBC/GBA LCD panels. My personal preference is to apply settings to correct the oversaturation because I love my retinas.

For color adjustment and LCD display simulation:

Quick Menu
> Shaders
>> Load
>>> >Shaders_glsl
>>>> > handheld
>>>>> >**gba-color.glslp**
>>
>> >Shader Passes: **2**
>> >Shader #1 > shaders_glsl > handheld > shaders > **lcd3x.glsl**
>> >Shader #1 Filter: **Default**
>> >Shader #1 Scale: **2x**

Exception:

The **KT-R1 C** uses a 3:2 display, however the resolution is *not* an integer scale of the GBA’s native output resolution of 240x160. Integer scaling will not fill the entire display.

A couple notes on LCD grid shaders/filters:

There are a number of different ways to achieve this effect, however bear in mind:
- There are very few LCD filters/shaders (if any) that will scale cleanly unless integer scaling is used. A 720p or greater display is recommended if you want to use an LCD filter/shader while using the full vertical screen space, at the ‘correct’ aspect ratio, or the grid will appear uneven.
- You will need to experiment with different shaders and filters to achieve good results. Due to the wide variety of different outcomes for different displays, resolutions, aspect ratios, shaders, and video filters, I simply don’t have the energy to cover this subject with the attention it deserves.

GBC settings (Gambatte core)

The below settings are best for displays with a resolution of 640x480 or less.

Settings
  > Video
  > Scaling
    - Integer Scaling: **OFF**
Quick Menu
  > Shaders
    > Load
      > Shaders
        > Interpolation
          - Bandlimit-Pixel.glslp
        > Parameters
          - Smoothness: 2.0 (or 3.0, if scaling artifacts are visible)

A well known ‘quirk’ of GBC and (particularly, earlier) GBA titles, was the use of highly saturated colors to compensate for the desaturating effect of the GBC/GBA LCD panels. My preference is to apply settings to correct the oversaturation.

Quick Menu
  > Options
    > Color Correction: **GBC only**
      > **Accurate**
      > Above Screen (Note: This is strongly a matter of personal preference.)

EXCEPTIONS: For emulating the GB/C on displays with a resolution of 640x480 or less, As stated above: there are very few LCD filters/shaders (if any) that will scale cleanly unless integer scaling is used. A 720p or greater display is recommended if you want to use an LCD filter/shader while using the full vertical screen space, at the ‘correct’ aspect ratio.

NOTE: For replicating the original Gameboy DMG screen look The ‘Gameboy3x_DMG’ filter is remarkably accurate, however running this filter with the Bandlimit-pixel shader will cause performance issues on lower powered devices. If you *must* use this filter, it’s best to stick with integer scaling. However there are in my opinion ‘good enough’ alternatives available by adjusting the color settings in the Gambatte core options, and sticking to the LCD**2x** video filter (not the shader!)

For lower powered Devices with 640x480 displays (inc. Miyoo Mini, and RG35xx caveats.**)**
On devices running on older ingenic SOCs like the JZ4770 or other lower-power devices, some of the above options will not be available (particularly, shaders) or will harm performance. In these cases, your options will be limited to filters and other, simpler scaling settings.
If you're using a device on this chipset with a smaller resolution screen like the 280v/m, it's a bit of a struggle to get a clean picture, so you're likely to get the best results using bilinear filtering or integer scaling. That said, the bicubic artifacting is less noticeable on smaller screens; YMMV. Worth noting that SNES and CPS1/2/3 emulation will rarely run at full speed with any filters beyond bilinear/bicubic on JZ4770 devices.

SNES / CPS(1/2/3) (JZ4770)
Settings > Video > Scaling
>Integer Scaling: **OFF**
>Keep Aspect Ratio: **OFF**
> Image Interpolation: **Bilinear**
> Video Filter: **NONE** (Any of these will tank performance for SNES/CPS.)

Gameboy DMG/Color/Advance
Settings > Video > Scaling
>Integer Scaling: **OFF**
>Keep Aspect Ratio: **ON**
> Image Interpolation: **Bicubic***
> Video Filter: ***Grid3x*** (for nostalgia) or ***Normal4x*** (for clarity)
- For **GBA**, use ***Grid2x*** or ***Normal2x***

NES / TG16(CD) / NeoGeo
Settings > Video > Scaling
>Integer Scaling: **OFF**
>Keep Aspect Ratio: **OFF**
> Image Interpolation: **Bilinear**
> Video Filter: **Normal2x**
- For **NeoGeo,** use **Upscale_256x-320x240**

Sega Genesis
Settings > Video > Scaling
>Integer Scaling: **OFF**
>Keep Aspect Ratio: **OFF**
> Image Interpolation: **Bicubic***
> Video Filter: **Upscale1.5x** (any higher and you start taking performance hits.)

*Bicubic filtering artifacts are reduced by upscaling; provides a sharper image than bilinear.

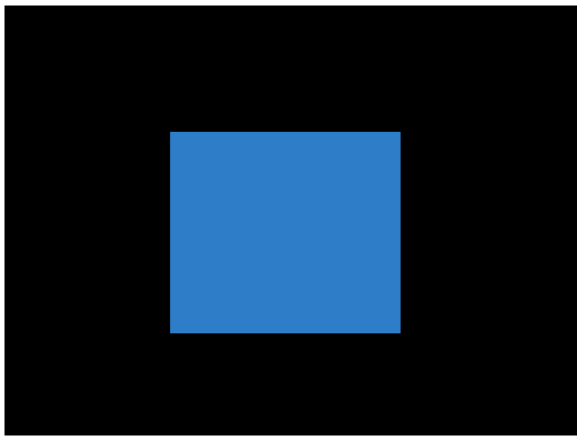
A NOTE ON THE MIYOO MINI AND RG35xx:
This devices are beefy enough to handle the normal3x filter and bilinear filtering enabled without a performance hit in almost all cases. PSX titles may suffer a bit, however for all others, it looks pretty good. The Miyoo Mini can take this up to 4x for all but PSX.

So, why these settings, specifically?
There are a number of differing opinions regarding what the 'best' possible image for a game will be. My goal with these settings is to:
- Get the sharpest possible image While using as much of the screen space as is sensible,
- Without introducing blur, scaling/sharpening artifacts, or other unpleasant 'smooth' filtering,
- At the 'intended' aspect ratio for each given system or game, which is not always native PAR.
- Which is useful to determine a baseline image expectation.

Achieving these goals on modern displays for 8, 16, and 32 bit titles can be tricky. Any given game title will have been made at a set internal aspect ratio and resolution. But the CRT displays these games were originally designed to be played on will display whatever is pumped into their inputs at the *display's* aspect ratio - not at the AR of the game content itself. So, most system developers of the day had little reason to consider the 'raw'/native aspect ratio of the resolution they were working at, except to compensate for the difference in the games' internal Display Aspect Ratio (DAR), and the CRTs literal dimensions (or in some cases, manipulate assets to play nice with CRT artifacts.) Developers were not, however, always consistent about their approach to this as there were few industry standard development resolutions beyond what worked for the developers.

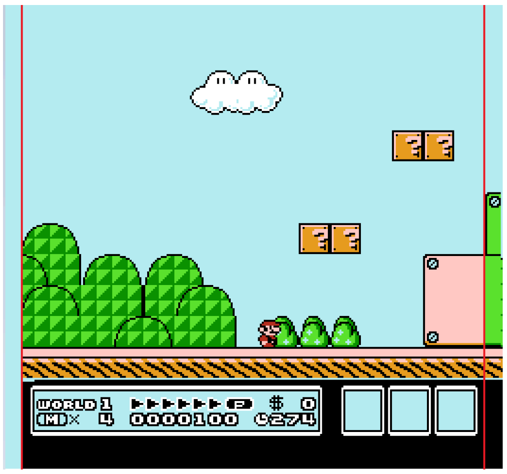
This results in a phenomenon where, despite the fact that most CRT displays of the time were of of a 4:3 (or somewhat less commonly 5:4) aspect ratio - you still see games with native Pixel Aspect Ratios (PAR) of 8:7 (SNES, Genesis) 10:7 (Genesis, NeoGeo), 12:7 (CPS1/2/3) - with quite a few others being used by more obscure platforms. But the point here is to demonstrate, the 'internal' image geometry does not have a 1:1 relationship with that of the 'external' display, and so very few games' native internal resolution will scale 'cleanly' on modern displays - even and particularly, the oft cited 640x480/4:3 'golden standard'. There are multiple reasons for this, which make more sense if we define three different types of aspect ratio:
1 - "Pixel Aspect Ratio" or **PAR** - This is the aspect ratio of each individual pixel. While this is mostly an incidental quality, Retroarch allows you to set the PAR to 1:1. More on this later.
2 - "Display Aspect Ratio" or **DAR** - aspect ratio of the image to be displayed on the screen. This is the aspect ratio of the 'final' image, once it has been fully rendered.
3 - the native Aspect Ratio of the display itself, or just **AR**. This is the real world-measurement ratio of the length and width of the display. Most of the time this will be proportional to the DAR, but there are exceptions. (the RG280m/v, for example.)

Enter: Scaling. In the simplest sense scaling is what happens when you stretch an image out to avoid playing Video Games for Ants. As an example, Chrono Trigger for the SNES uses an 8:7 DAR (along with most SNES titles), at a resolution of 256×224. On a 640x480 display, with no scaling, you don't get a very large image:

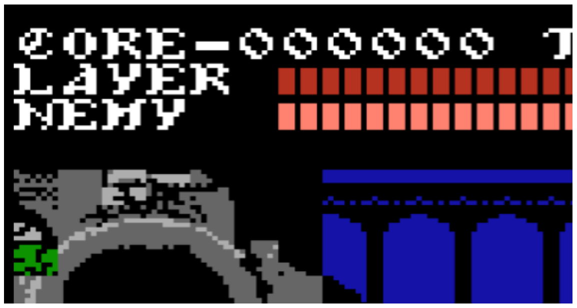


Notes on the limitations of 640x480 displays - And the NES overscan quagmire
Often touted as the 'gold standard' for classic gaming - and not entirely without good reason. The 640x480 resolution is usually high enough to minimize most scaling or stretching artifacts, while allowing a generous amount of the display space to account for a number of scaling/stretching preferences.

But it's not without limitations: it's commonly believed that SNES, Genesis, and other consoles of the day were 'designed' for this resolution, but this is unfortunately false. In many cases, the console designer's primary concern in regards to output resolution was to utilize a resolution high enough to allow for good visual fidelity without needing to employ overly expensive hardware to achieve this. The NES made use of a 256x240 resolution - an unusual 16:15 AR (which does not scale cleanly to 640x480, for obvious reasons.). However, this doesn't account for the fact that some display areas wouldn't be visible on most CRT displays; as they live in the overscan area - space that would ordinarily be drawn outside of a CRT's vertical or horizontal range wjocj normally contain nothing, or garbage visual data You can see this in in Super Mario Bros. 3. Where a column of 8 pixels straddles either side of the image. By default, you'll see a column of noisy visuals on the right, and a blank column on the left. (above, delineated in red)



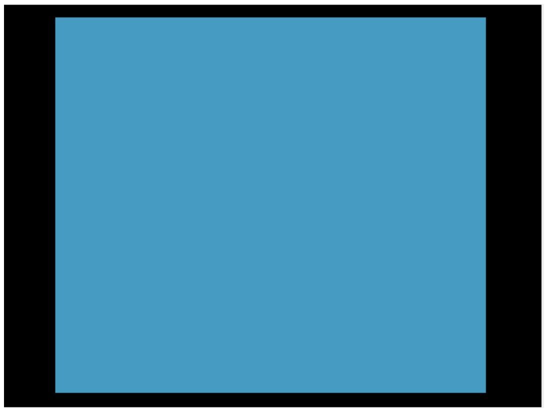
If you want to do more reading on what why this space existed, research terms: *Vblank, Hblank*



This throws the notion of clean 640x480 into further disarray - since not all games used the overscan space as anything other than part of the display. That same left column in Castlevania, for example, contains part of the HUD, and hiding this portion of the overscan (A feature of most NES emulators) truncates the text to confusingly, refer to CORE, LAYER, and NEMY. While this is a quirk unique to the NES, no two games necessarily use the overscan space in the same fashion, so there is no universal setting that works best for all.

Best advice I can give is to be willing to tinker, or adopt a mindset towards overscan artifacts that allows you to find zen in the face of this overwhelming visual obnoxiousness. And good on you if you do - I'm personally way too neurotic about this shit and ***will never get over it.***

Notes on Integer scaling
(1:1 PAR and why many people recommend it)

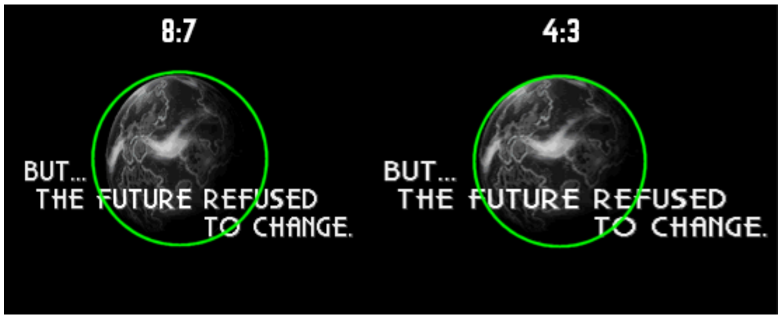


Integer scaling is a scaling method that simply multiplies the X and Y pixel-dimensions of the image by a fixed, whole number. PAR, or 'Pixel Aspect' ratio is the aspect ratio of each pixel on the screen - 1:1 assumes perfectly square pixels. Most LCD-based displays have a native PAR of 1:1. So a 256x224 image at a 2x integer scale, becomes a 512x448 image. Not quite 640x480 but a much more practical viewing space. It will however still be at 8:7 DAR, which will not fill the entire display. The main advantage to 1:1 PAR integer scaling is the very sharp, 'clean' look. Every pixel is a perfect square and aligns with the pixels of the display itself - so there is no distortion or blurring, as one might see with other scaling or filtering methods. Another advantage is that the processing overhead for this method is virtually nonexistent.

"Okay but….what about Integer scaling at PARs other than 1:1?"
There is no reason to do this. If you change the DAR to anything other than a game's native 1:1 PAR while also using integer scaling, your image is no longer truly integer scaled. So you get the disadvantages of both integer scaling (lost screen space; possible bad geometry) AND filters (Distortion) - with the advantages of neither. It is possible to effect this on exceptionally high resolution displays (1440p and up) without artifacting, but it's a pain to set up.

<u>There are two main downsides to 1:1 PAR Integer scaling:</u>
1 - The obvious one - You sacrifice some screen space. Since the native resolution of the image doesn't scale neatly to the 640x480 display, you get a black border around the image.

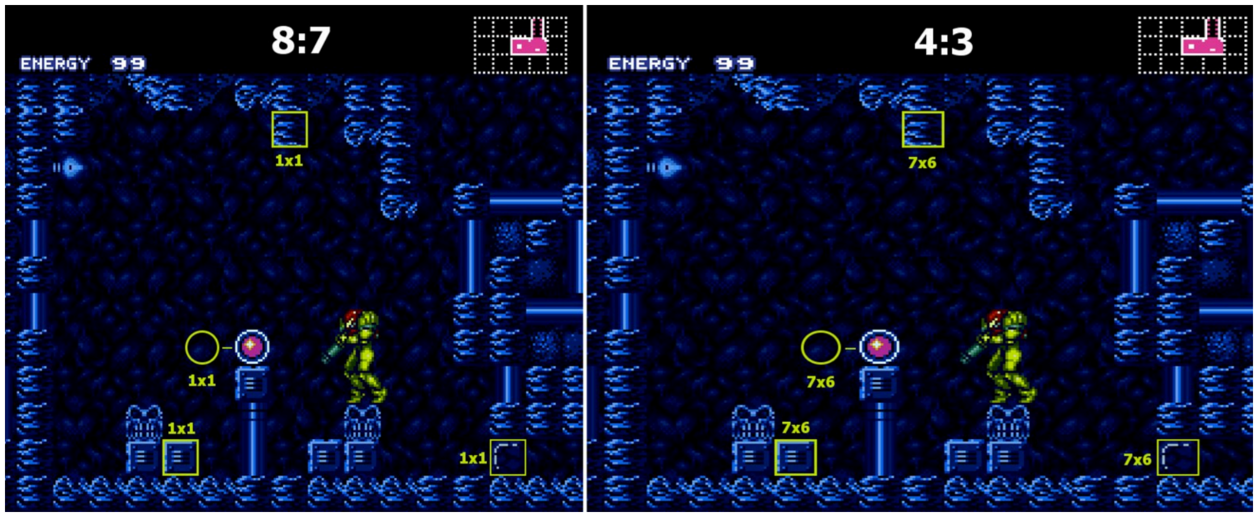
2 - The less obvious, but in my opinion, more important one:
Many games of the 8-to-32 bit eras were designed to be played on 4:3 displays, with art assets made to compensate for the difference between the system's native output and that of the target display. An easy example of this in action, comes (again) from Chrono Trigger. CT displayed in its native 8:7 results in the geometric distortion of certain images



In the 'native' example to the left, we can see that the globe is slightly narrower than the circle that's been drawn around it. In 4:3, we see where the globe fills the circle accurately.

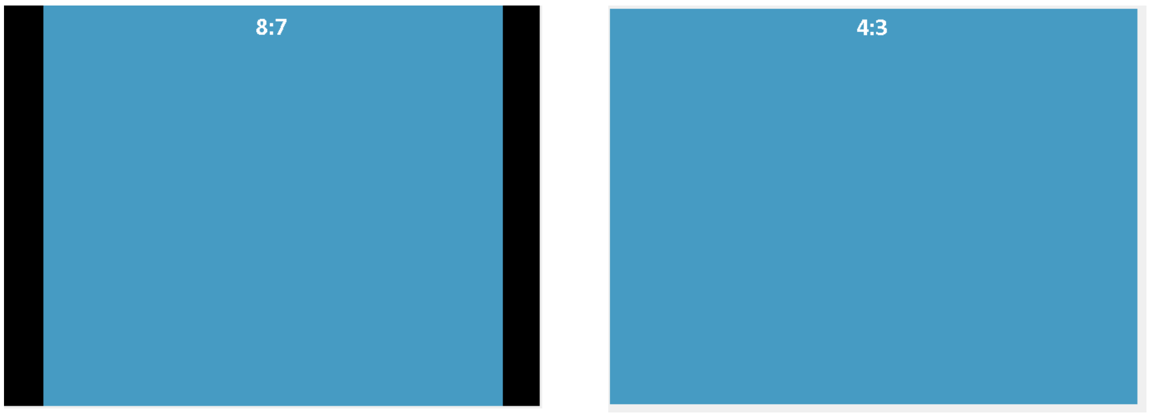
While the ‘circle test’ is not aways (nor should it be) the sole determinant for a game’s intended aspect ratio, it is a useful quick-reference. There are some titles where this is even an exceptionally bad benchmark, because circular objects are not consistently circular; such as Castlevania: SotN. There are many other examples in other games, but I won’t bore you with all the ones I know of - I think you get the idea.

Amusingly - a lot of games were *not* designed with this compensation in mind, including Super Mario World and pictured here - Super Metroid:

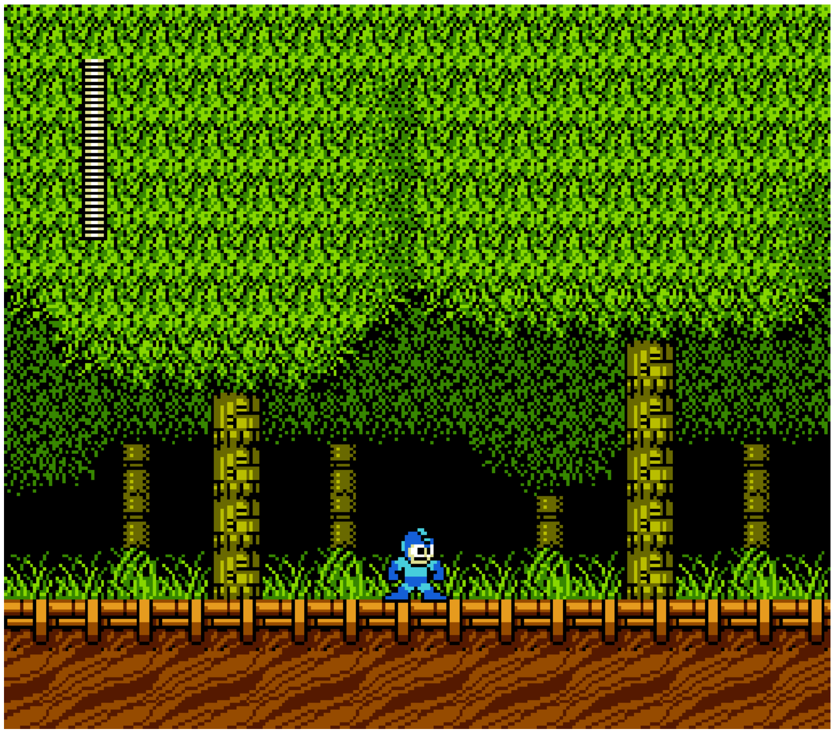


Which of these was intentional, or even just looks better is obviously debatable - but at the very least we have some visual evidence that game devs were not consistent about whether or not their work compensated for the difference in display and native aspect ratios.

ENTER: Non-Integer scaling
This is where the raw image is scaled using fractional, rather than whole numbers. In most cases, turning integer scaling off will adjust the image to fill as much of the screen as possible while maintaining the defined DAR, per the examples below:

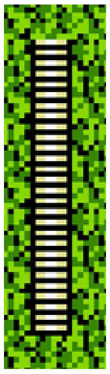


This is where filters and shaders become a factor. While we've expanded the image to fill the display (or as much of the screen as possible, given the DAR we've set.) we now have the problem of everything looking...kind of janky:



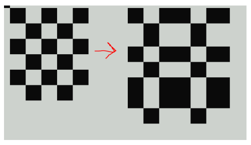
Nearest Neighbor Filtering

The image to the left is being displayed using 'nearest neighbor' filtering. It may seem initially fine as a still picture but if you look closely, you can see the distorting effect of non-uniform pixels on Megaman's life bar:



This is caused by the system approximating rows/columns of pixels where the resolution of the image does not evenly match that of the display, which results in the output image suffering from columns and rows being rendered with non-uniform widths and heights..

You might see this effect called 'pixel distortion', or when in motion, 'wobble' or 'shimmer'. *The lower the resolution of the display, the more pronounced these artifacts will be.* A more extreme example might look like the image to the right:



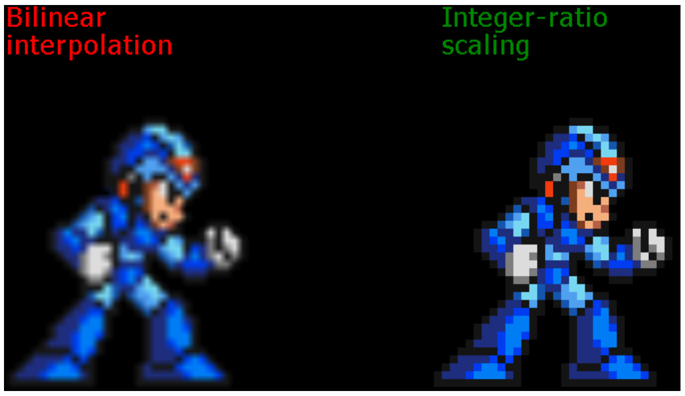
So how do we compensate for these new problems?

Filters and Shaders! If you want my recommendations for filter and shader settings, scroll back up to page 2 of the document. This will eliminate almost all visible distortion and shimmer, while still providing a very crisp image. But for the sake of being thorough, I will go over some of the more common filtering methods:

Bilinear / Bicubic/ Trilinear Filtering

These filtering types are handy, low impact and some of the most common methods of addressing pixel distortion on non-integer scaled images. However, The end results are 'soft' looking, and while bicubic/trilinear filters reduce the fuzz, they leave sharpening artifacts.

An example of bilinear filtering:



I'll add Bicubic and bilinear here at some point…

Finding a comfortable medium

Retroarch provides a number of shaders and filters that allow you to further manipulate the image to reduce the distortion artifacts of non-integer scaling, as well as sharpen the image to something more visually appealing than the blurry results of bilinear/trilinear/bicubic interpolations.

The difference between Shaders and Filters:

The technical specifics are complicated, but the end result is that they're both different methods of applying image enhancement, that can be combined with one another for various results.

Shader recommendations

***Interpolation* -** Shaders in this group are designed to mitigate visual artifacts introduced by non-integer scaling, or to smooth the image for people who aren't into the chunky, pixelated look. In recommended order:

- **Bandlimit-Pixel:**	Eliminates almost all blurring and'wobble'. Somewhat resource hungry. Adjustable blur to tailor to personal Preferences and to reduce what little wobble remains.
- **Quilez** :	Eliminates most wobble and pixel distortion; less effective than Bandlimit-Pixel, but lower performance impact.Slightly blurry though less blurry than bilinear filters. Low overhead.
- **Sharp-Bilinear-2x-Prescale:**	Mitigates blurring, but leaves some 'wobble' artifacting. Low resource consumption, so can be useful if you Experience poor performance using other shaders.
- Smoothstep / Smuberstep:	I can't tell much difference between these, but they're a sort of midpoint between Sharp-Bilinear-2x-Prescale And Quilez with similar overhead.

NOTE: depending on the retroach version and device, the above shaders may be in the 'pixel' folder.

***Handheld* -** Shaders that emulate the look of LCD screens on various handheld systems

- LCD3x: A nice 'grid' that emulates the look of older handheld gaming systems. Primarily useful on handhelds with a native AR equal to or wider than 3:2; may experience artifacting on 480p displays.

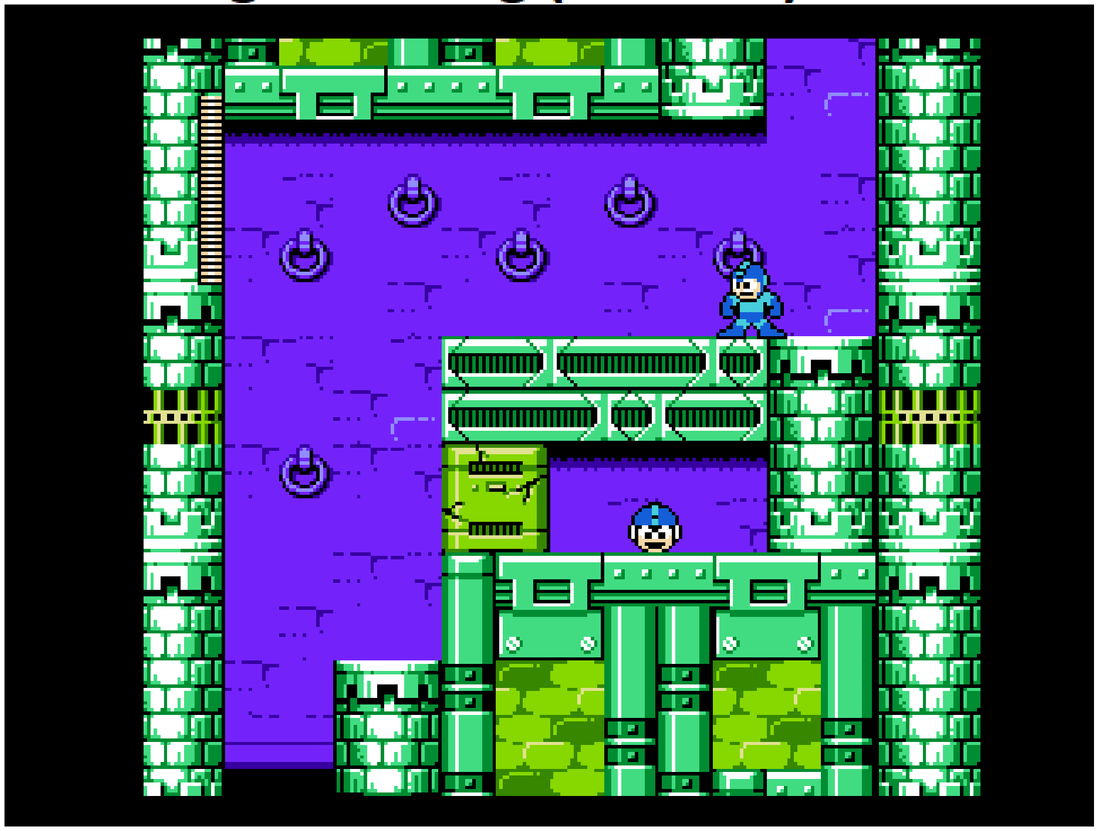
- GBA-Color - Useful to mute the garish colors of GBA, NGPC, and Wonderswan Color titles. Many developers purposefully manipulated the palettes of games for these systems to compensate for the darkening/desaturating effects of their screens, which results in a picture that, at least in my opinion, looks awful. This shader reduces the eye-burn effect, and will look more accurate to the original systems. Most GB/GBC/GBA emulators include this as a feature.

Filter/Shader Comparison Screenshots

- I took these from my PC, using an arbitrary window size. I know it isn't scientific, but I couldn't find any option in RA to set a specific window resolution. I opted for a window AR that is close to 4:3, but it isn't precisely so. This is to give you an *idea* of what the various filters and shaders will do and look like on your handheld. I fully admit that this is not the most accurate way to go about it, but if you want to take the time to make some better images, I'll be happy to add them to this guide, with attribution.

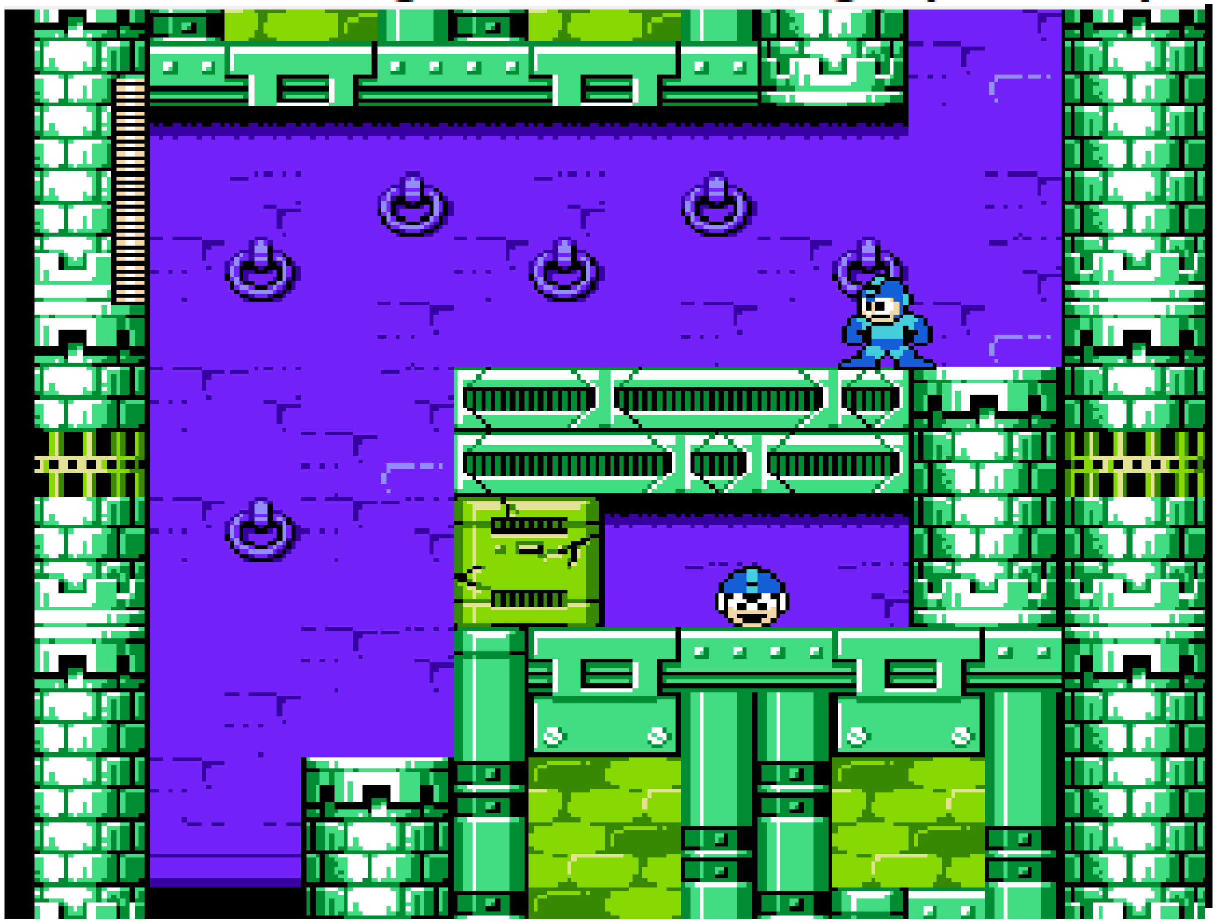
Two things to keep in mind:
1 - These images are blown up from a significantly smaller image, and thus will appear slightly more blurry in the document than all of their real-life examples.
2 - If you're viewing this document on a PC, the image will be substantially larger than you'd see on any 3.5" - 5" screen size handheld device. I didn't try to compensate for this, because I wanted to keep honest about the shortcomings of non-integer scaling.
3 - A lot of people won't be able to tell much difference between some of these options, and that's fine. Again, the settings that make you happy are the best settings!

Integer Scaling (1:1 PAR)



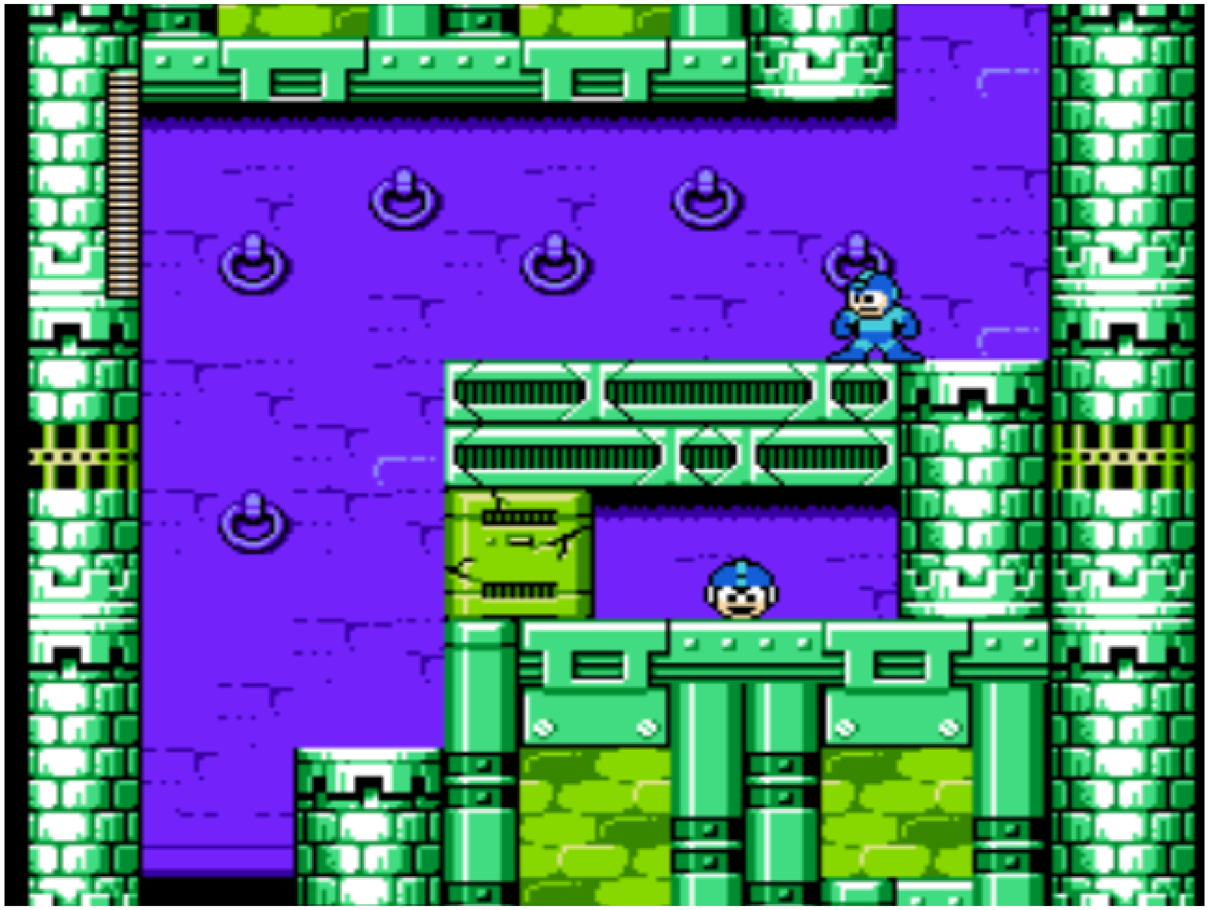
As described previously, this provides a very clean image, but leaves a lot of unused screen space. If you're accustomed to playing on 4:3 CRT displays, this probably looks slightly too narrow - being at an 8:7 AR. Many (if not most games) assets were designed to compensate for the fact that they would have been stretched to 4:3.

Nearest-Neighbor "filtering" (4:3 AR)



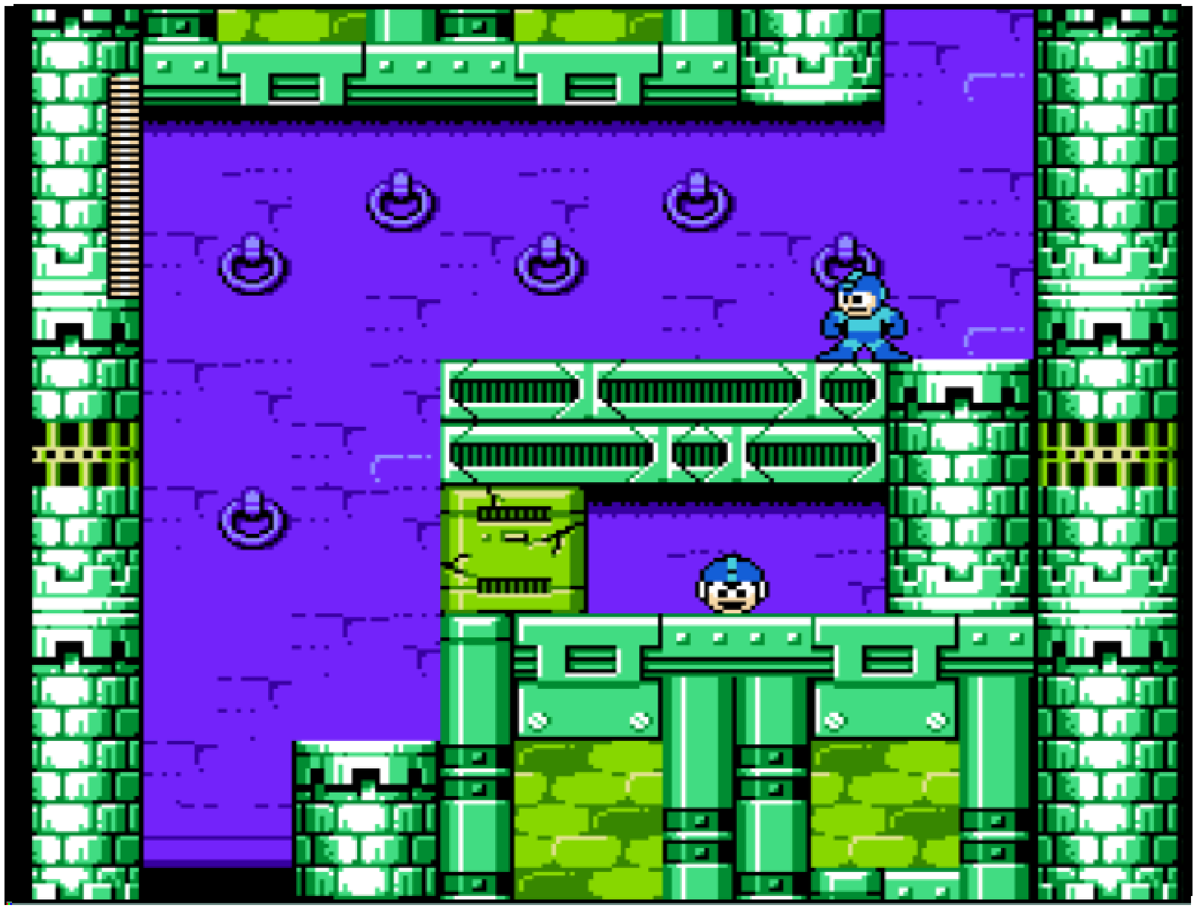
Here we fill up much more of the available screen space (on an actual 4:3 display, this would be ‘all of it’) However, the scaling artifacts are big, obvious and ugly when you look at Megaman’s life bar, or the grille on the platform he’s standing on. The mix of fat/skinny/tall/short pixels looks hideous.

Bilinear Filtering (4:3 AR)



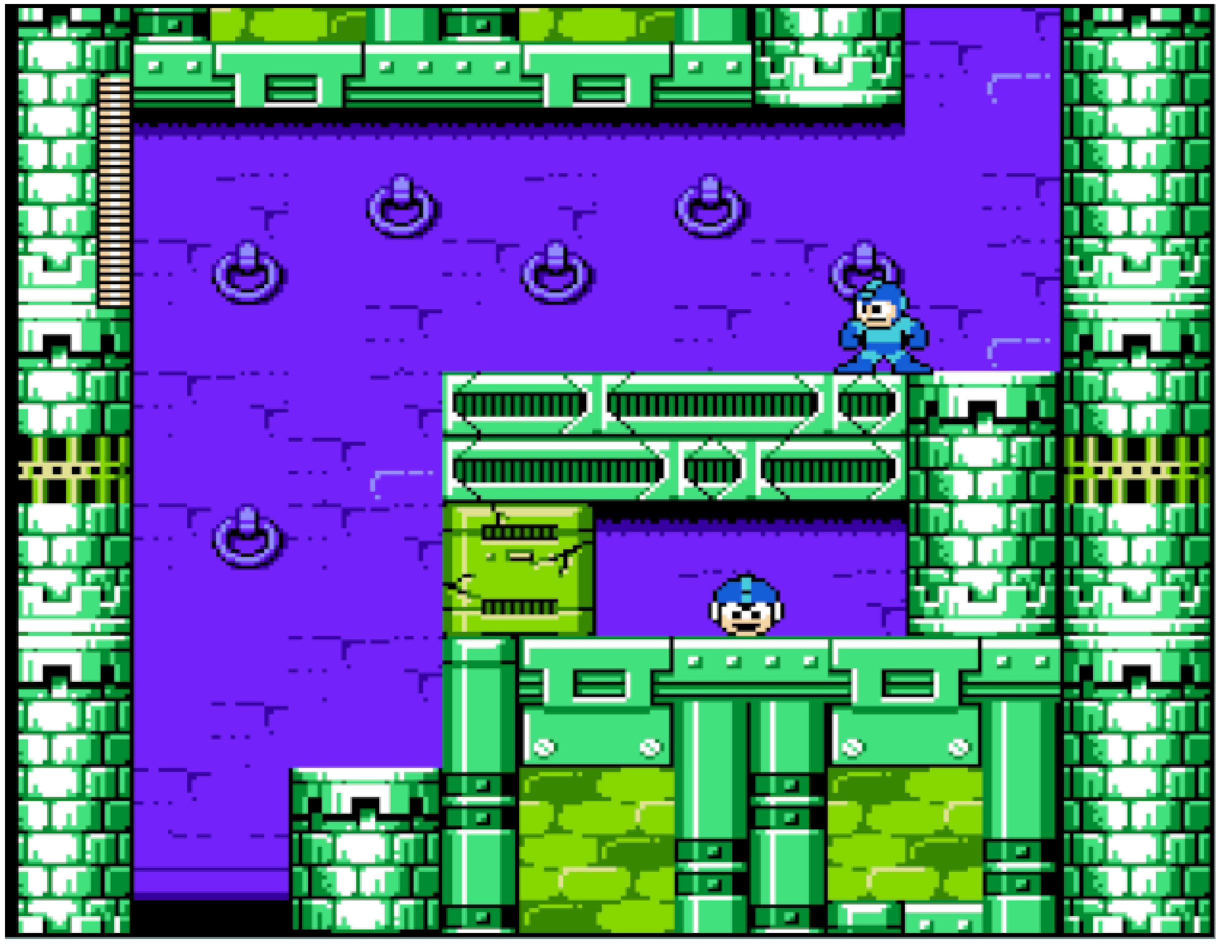
The filter eliminates most of the distortion by blurring everything together. Some people like this softer look, but I find it a bit muddy for my tastes.

Shader: Sharp-2x-Bilinear-Prescale (4:3)



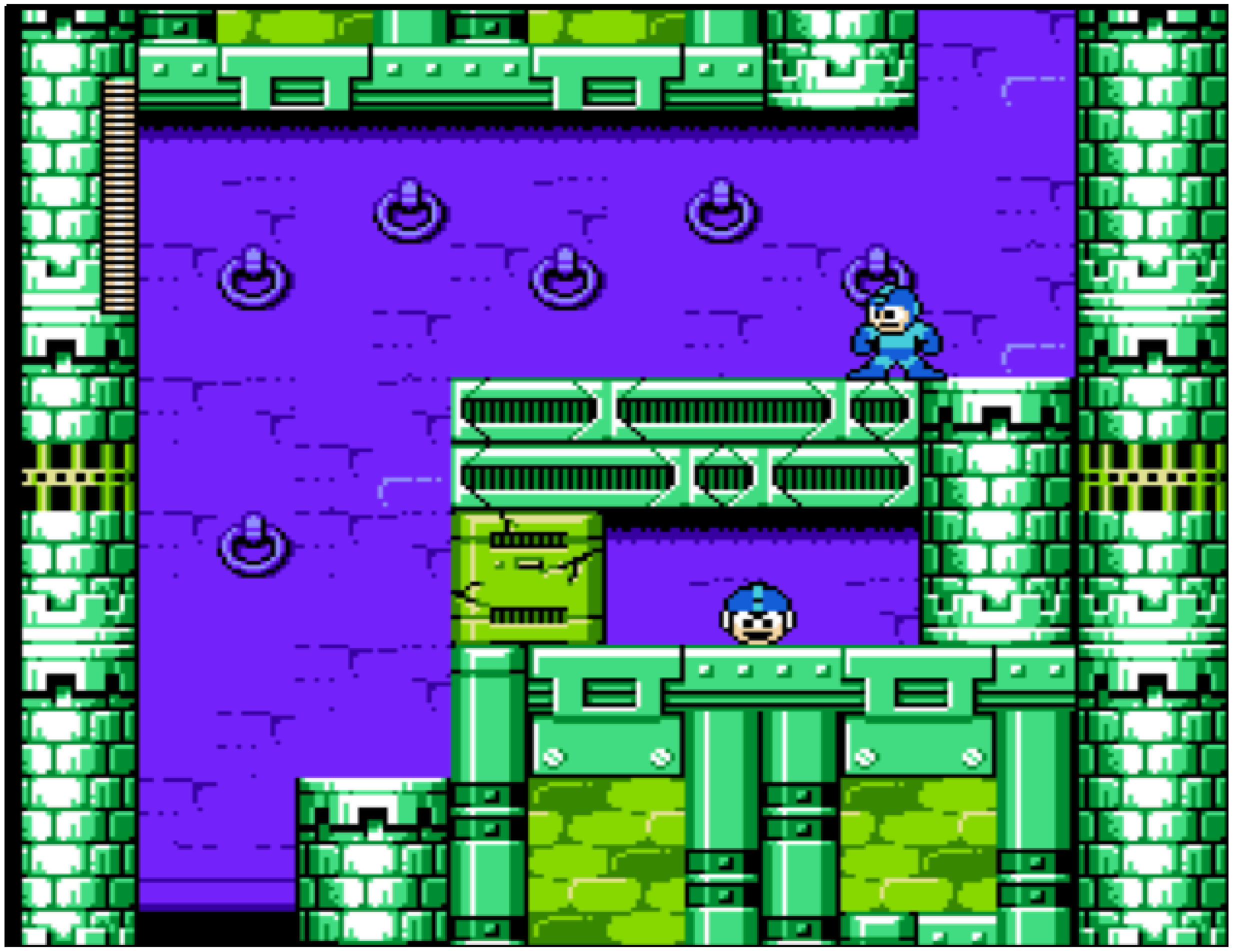
While this shader eliminates *most* of the distortion and blurring, both are still present if you look closely. It is most evident when looking at Megaman's Life bar. Games that feature heavy dithering will readily bring out the flaws of this shader.

Shader: Bandlimit-Pixel (Smoothing: 2.0)(4:3 AR)



NOTE: The above caption contains a typo: The ‘2.0’ should be ‘0.2’
Eliminates nearly all pixel distortion, with some - albeit very little - visible blurring compared to any of the above solutions. Though, again - if you’re viewing this document on a PC, the image is going to be much larger than any RK3326 handheld, so while some blurring artifacts are visible here, they are borderline imperceptible on a 3.5” - 5” display. As a side note, adjusting the ‘smoothness’ shader parameter will allow you to further balance the blur/distortion ratios. It’s not a perfect solution but it’s a damn fine compromise.

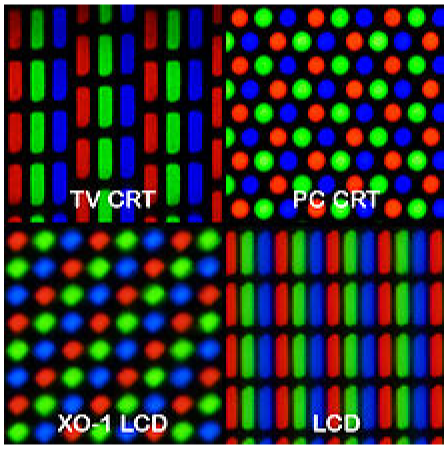
Shader: Quilez (AR 4:3)



Just a touch blurrier than Bandlimit-Pixel, but overall similar results with less overhead. If you find certain games performing badly while using Bandlimit-Pixel, try this one for a potentially better experience.

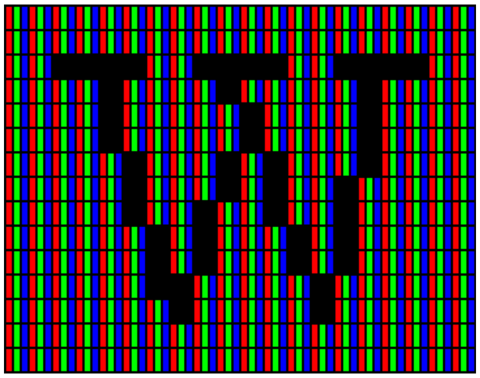
(To be added: Smoothstep/Smuberstep)

Some technical information regarding LCD displays and pixels *(Added 12/9/2021)*
While most people using handheld devices (or have used LCD-based displays in the now-decades they've been prominent) are likely already familiar with this, I think some of the above might make more sense if it is explained why the image behaves the way it does on LCD screens.

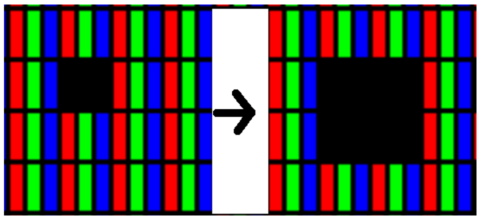


LCD screen pixels are (mostly!) square, with a PAR of 1:1. However, each pixel is made up of a single red, green, and blue LCD 'subpixel'. Different displays may use different subpixel formats, with the most common subpixel format being "RGB" The image to the right shows some common subpixel layouts, with the devices we're discussing using the listed "LCD" example. The vividness and color of each pixel is manipulated by altering the values of its constituent subpixels.

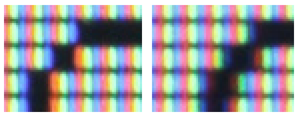
One of the reasons that many CRT displays have a somewhat 'fuzzy' look, is that the subpixel aperture grid is not a square grid.



What this means for your image
Because each screen pixel of the image may not line up neatly with the grid of pixels on your display, the image has to be scaled. Integer scaling, as previously detailed, simply blows the image up by a fixed, whole number multiplier (usually 2x or 3x, depending on the native resolution of the image) The "W" to the left is displayed at its native resolution, with each pixel of the image, occupying 1 pixel on the display. A 2x integer scale makes each source pixel occupy 4 pixels of the display.



Of course, we can't always do this and also have the image fill the entire display, so various methods of filling the gaps have to have been engineered. A black pixel can't just be expanded to occupy neighboring subpixels without creating chromatic aberrations or 'color fringing'. With no filtering, the image stretches; occupying whole pixels of the display.



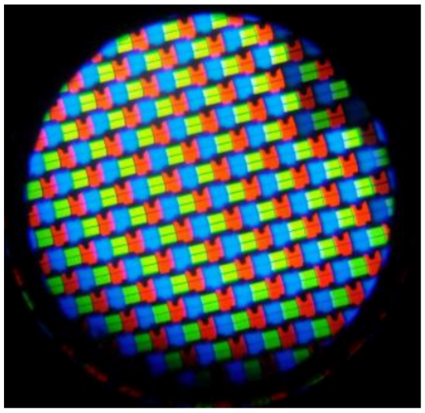
Color fringing is most often seen with 'cleartype' text. This is a type of anti-aliasing used to smooth the pixelated look of certain fonts, which dims/brightens a neighboring subpixel. This results in text having a multicolored, somewhat nauseating 'halo'. Many shaders and filters have been made as means of addressing these sorts of artifacts.

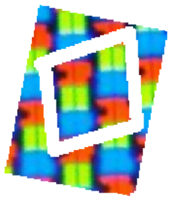
For some Information on what nearest-neighbor scaling/interpolation actually is and why it looks bad, check out - https://en.wikipedia.org/wiki/Texture_filtering#Nearest-neighbor_interpolation

The RG280m/RG280v - (Added 01/05/2022)

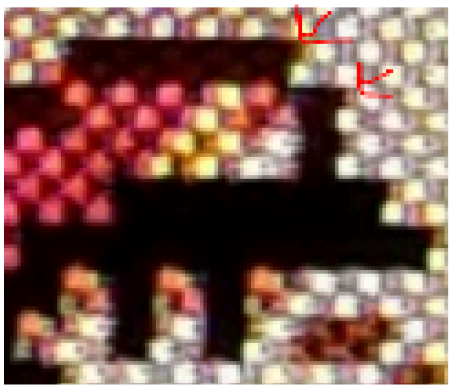
(Thanks to user 'Gabraham' in the RG Handhelds discord for the closeups, and sweet unintentionally synthwave samus; user SML from the Dingoonity forums for Mario's very flattering closeup shot)

The smaller anbernic handhelds in the RG280 lines make use of a unique (and in my opinion) interesting subpixel layout quite unlike most other LCD panels. It is well known at this point that RG280 devices have an unusual PAR. However the actual screen aspect ratio is still 4:3 despite being a 480x320 display, but because single pixels are much wider than they are tall, the 'functional' resolution is closer to 240x320, as the images are doubled in height to compensate for the squat pixels.



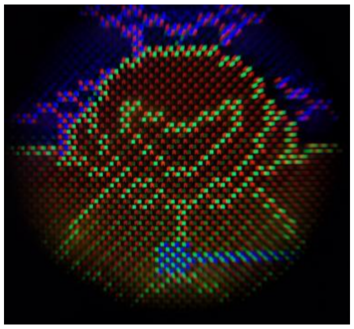


Thanks to Gabraham, we have some images of what this screen looks like up close and it reveals that the subpixel layout of the display is not only unusual in shape, but also an unusual configuration:



The pixels are made up of 3 subpixels like most LCD displays - but in a non-RGB format (RBG? GRB? Someone let me know what the correct naming here should be.) and a staggered subpixel grid. Since the input image needs to be doubled in height to avoid a terrible, squatty appearance, this gives the 'final' display pixels an almost rhomboid appearance. While the 2.8" screen is small enough to make this less noticeable to most, when zoomed in we can see the artifacts of this layout more clearly.

There is the screen door effect, and the corners of the pixels look a little bit…drunk due to the staggered nature of the grid. Though, speaking personally - this can look pretty awesome in some contexts. Using Synthwave Samus' closeup to the right - vertical and horizontal lines suffer a bit, the '/ 'diagonals are quite sharp. '\' diagonals break up a bit, which gives this a somewhat hatched appearance.



Thanks for reading.
Nerd

Changelog:

01/27/2022
- Figured it might be handy for my personal reference to start keeping track of changes and corrections. So this changelog is here, now.
- Moved all settings sections together. Previously, system specific settings were near the end of the document, and JZ4770 device retroarch settings even further in. These were moved to near the beginning for better organization.
- TOC updated to reflect new page numbers.

11/21/2022
- Added a section explaining why the guide exists
- Removed the 'device index' section from the TOC (Too many devices to keep meaningful track of)
- Added section detailing common screen aspect ratios and resolutions for popular handheld devices
- Working on removing language specific to the RK3326 chipset, since most of the information here is no longer specific to it.

07/26/2024
This guide is now pushing three years old! I've made a few stylistic changes and made corrections to recommended bandlimit pixel shader settings - 'smoothness' should be set to 0.2 in most cases, not 2.0. The latter makes the image extremely blurry.

08/30/2024
Removed some incorrect information about filters and shaders; edited the first 15 pages or so for style and clarity.