stb

single-file public domain (or MIT licensed) libraries for C/C++

Most libraries by stb, except: stb_dxt by Fabian "ryg" Giesen, stb_image_resize by Jorge L. "VinoBS" Rodriguez, and stb_sprintf by Jeff Roberts.

library	lastest version	category	LoC	des
stb_vorbis.c	1.16	audio	5486	decoc vorbis file/me float/1 signe
stb_image.h	2.22	graphics	7547	image loadin from file/me JPG, BMP, HDR,
stb_truetype.h	1.21	graphics	4882	parse and ra chara truety
stb_image_write.h	1.13	graphics	1622	image disk: F BMP
stb_image_resize.h	0.96	graphics	2630	resize larger with g
stb_rect_pack.h	1.00	graphics	628	simple rectar with d quality
stb_ds.h	0.4	utility	1669	typesa dynan and h for C, compi

stb_sprintf.h	1.06	utility	1860	fast sp snprin C/C++
stretchy_buffer.h	1.03	utility	262	typesa dynan for C (appro vector doesn as C+
stb_textedit.h	1.13	user interface	1404	guts c editor etc im them scratc
stb_voxel_render.h	0.88	3D graphics	3806	Minec voxel "engir many featur
stb_dxt.h	1.08b	3D graphics	728	Fabia Giese time D compr
stb_perlin.h	0.4	3D graphics	366	revise noise 1D ou
stb_easy_font.h	1.0	3D graphics	303	quick- easy-t bitma printin rate, e
stb_tilemap_editor.h	0.41	game dev	4161	embe tilema
stb_herringbone_wa...	0.7	game dev	1221	herrin Wang gener
stb_c_lexer.h	0.10	parsing	964	simpli parse

				langu
stb_divide.h	0.92	math	421	more bit mo "euclid divide
stb_connected_comp...	0.96	misc	1049	increm compu reach grids
stb.h	2.34	misc	14453	helper for C, redun C++; l autho perso
stb_leakcheck.h	0.5	misc	190	quick- malloc check

Total libraries: 21
Total lines of C code: 55652

FAQ

What's the license?

These libraries are in the public domain. You can do anything you want with them. You have no legal obligation to do anything else, although I appreciate attribution.

They are also licensed under the MIT open source license, if you have lawyers who are unhappy with public domain. Every source file includes an explicit dual-license for you to choose from.

Are there other single-file public-domain/open source libraries with minimal dependencies out there?

Yes.

If I wrap an stb library in a new library, does the new library have to be public domain/MIT?

No, because it's public domain you can freely relicense it to whatever license your new library wants to be.

What's the deal with SSE support in GCC-based compilers?

stb_image will either use SSE2 (if you compile with -msse2) or will not use any SIMD at all, rather than trying to detect the processor at runtime and handle it correctly. As I understand it, the approved path in GCC for runtime-detection require you to use multiple source files, one for each CPU configuration. Because stb_image is a header-file library that compiles in only one source file, there's no approved way to build both an SSE-enabled and a non-SSE-enabled variation.

While we've tried to work around it, we've had multiple issues over the years due to specific versions of gcc breaking what we're doing, so we've given up on it. See https://github.com/nothings/stb/issues/280 and https://github.com/nothings/stb/issues/410 for examples.

Some of these libraries seem redundant to existing open source libraries. Are they better somehow?

Generally they're only better in that they're easier to integrate, easier to use, and easier to release (single file; good API; no attribution requirement). They may be less featureful, slower, and/or use more memory. If you're already using an equivalent library, there's probably no good reason to switch.

Can I link directly to the table of stb libraries?

You can use this URL to link directly to that list.

Why do you list "lines of code"? It's a terrible metric.

Just to give you some idea of the internal complexity of the library, to help you manage your expectations, or to let you know what you're getting into. While not all the libraries are written in the same style, they're certainly similar styles, and so comparisons between the libraries are probably still meaningful.

Note though that the lines do include both the implementation, the part that corresponds to a header file, and the documentation.

Why single-file headers?

Windows doesn't have standard directories where libraries live. That makes deploying libraries in Windows a lot more painful than open source developers on Unix-derivates generally realize. (It also makes library dependencies a lot worse in Windows.)

There's also a common problem in Windows where a library was built against a different version of the runtime library, which causes link conflicts and confusion. Shipping the libs as headers means you normally just compile them straight into your project without making libraries, thus sidestepping that problem.

Making them a single file makes it very easy to just drop them into a project that needs them. (Of course you can still put them in a proper shared library tree if you want.)

Why not two files, one a header and one an implementation? The difference between 10 files and 9 files is not a big deal, but the difference between 2 files and 1 file is a big deal. You don't need to zip or tar the files up, you don't have to remember to attach *two* files, etc.

Why "stb"? Is this something to do with Set-Top Boxes?

No, they are just the initials for my name, Sean T. Barrett. This was not chosen out of egomania, but as a moderately sane way of namespacing the filenames and source function names.

Will you add more image types to stb_image.h?

If people submit them, I generally add them, but the goal of stb_image is less for applications like image viewer apps (which need to support every type of image under the sun) and more for things like games which can choose what images to use, so I may decline to add them if they're too rare or if the size of implementation vs. apparent benefit is too low.

Do you have any advice on how to create my own single-file library?

Yes. https://github.com/nothings/stb/blob/master/docs/stb_howto.txt

Why public domain?

I prefer it over GPL, LGPL, BSD, zlib, etc. for many reasons. Some of them are listed here: https://github.com/nothings/stb/blob/master/docs/why_public_domain.md

Why C?

Primarily, because I use C, not C++. But it does also make it easier for other people to use them from other languages.

Why not C99? stdint.h, declare-anywhere, etc.

I still use MSVC 6 (1998) as my IDE because it has better human factors for me than later versions of MSVC.