CXBX-R & CE QUICKSTART GUIDE by ycarcomed

v3 October 2020

1. Which build?

Cxbx-r is in active development, and each build might behave differently for JSRF. Some significant points:

- ~March-April 2018: Boost dash no longer a black screen. Heavy performance hit.
- January 2nd 2019: Upscaling support. Boost dash is all blur, unplayable again.
- 2020: Boost works again. Graffiti still broken. Unplayable performance hits on AMD CPUs.
- 2021: No more performance issues on AMD CPUs. Graffiti broken.

~~I use Mar 19, before boost dash fix. I find it's the build that has been most stable for me and have had zero crashes with it. Fuse uses Mar 25.~~
My main build is Dec 6 2018. Available here.

To grab any build, go to https://cxbx-reloaded.co.uk/download. Builds earlier than April 2020 are not available on Github releases. The only way to acquire them is to clone the repo and build them yourself.

If even Dec 6 is too slow for you, **you can grab Mar 19 here** (the date says Nov 17, because that's the day I built it after I formatted my PC and lost it).

2. Save files and cache folder

Save files for JSRF are located at your
%appdata%\Cxbx-Reloaded\EmuDisk\Partition1\UDATA\5345000a\99271B32E8BB
The game is hard coded to load the file named **JSRFDATA.SAV**.
At https://www.speedrun.com/jsrf/resources you can find a selection of vanilla save files (manual renaming needed to use) or Fuse's quicksave mod with a selection of premade save slots all in one place (recommended).

The cache is at **%appdata%\Roaming\Cxbx-Reloaded\EmuDisk\Partition5**
What this folder does is emulate the location where the original hardware stores game assets on the HDD for faster loading. At load time, the game will first look for every (cacheable) file in

the cache folder. If it doesn't find it there, it will load it from "disc" i.e. the location where your xbe and game assets are. **Keep that in mind when modding files. If you edit just the original file, the game will just load the unedited cached file (you can just edit the cached file instead). The original and cached version of the file are not necessarily interchangeable (file size may change)! Don't lose track of your stuff! Keep backups!** Deleting any part of the cache will cause it to be rebuilt from game assets when it is time for the game to load them.

3. Controller

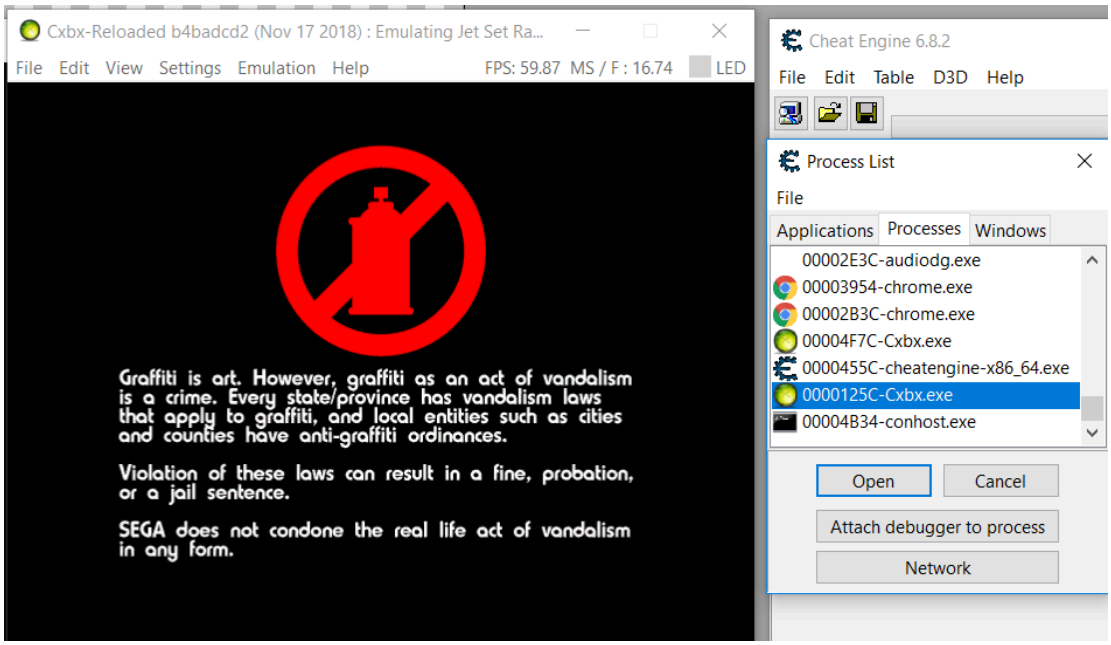
If your controller supports Xinput it should be the better choice. Otherwise use DirectInput and set up your mappings. Under DirectInput Cxbx-r does not support a hat switch, so your D-pad might not work. I use an original Xbox S controller with a homemade USB adapter and XBCD. In XBCD, I reprogrammed the D-pad as 4 digital buttons instead of a hat switch, so I can use it with Cxbx-r.

4. Using CE with Cxbx-r

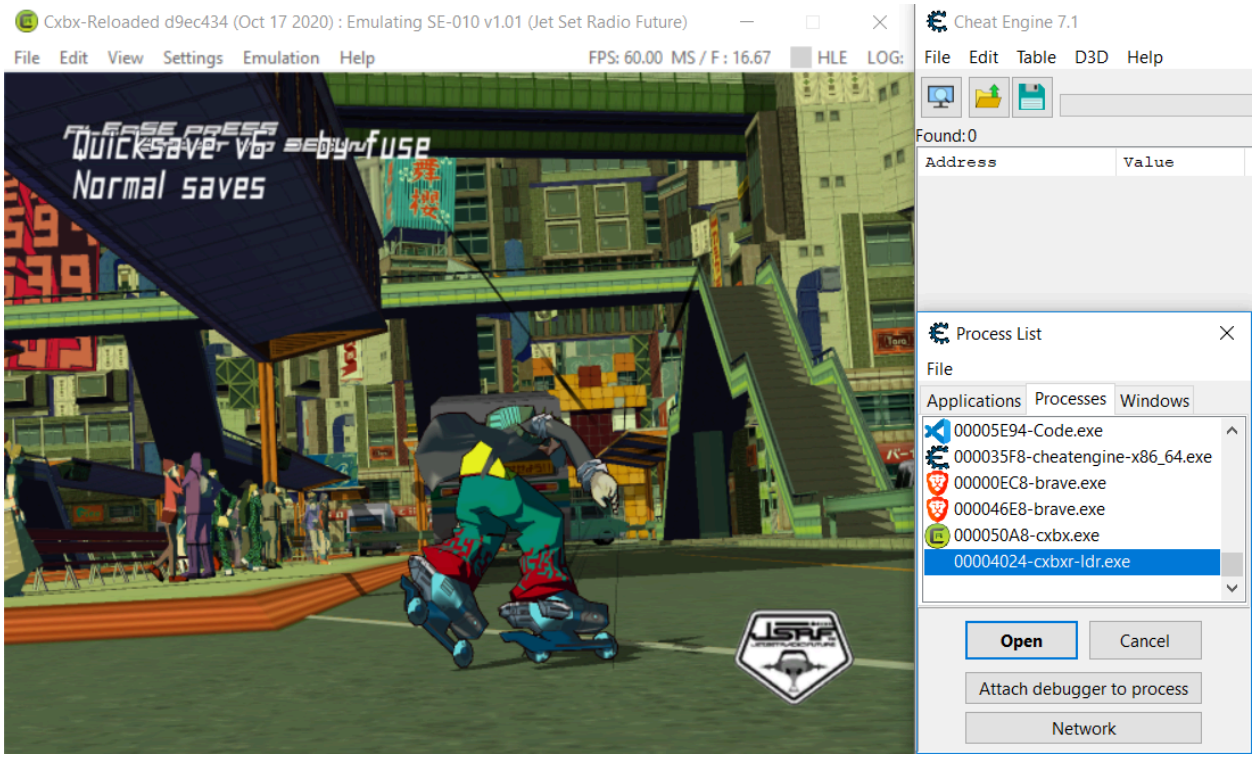
Cxbx-r runs in two separate processes, one for the UI and one for emulation. Thus you have to take care to hook CE to the correct process.

2019 and earlier

First start emulation (F5 key) then scroll down to the **second Cxbx.exe process** in CE's open process window.



2020 onwards



First start emulation (F5 key) then scroll down to **cxbxr-ldr.exe** in CE's open process window.

Note that you can automate the attachment to the desired process from CE's settings. Go to Settings > General settings, and in the "automatically attach to process" textbox input **cxbxr-ldr.exe;cxbx.exe**. The order is important to ensure it will work with both old and new builds.

Find the latest **jsrf.CT** (developed with NTSC-U JSRF. A less up-to-date NTSC-J version also exists) at JSRF Reverse Engineering Discord. I update it as new stuff is discovered. Dropdown groups are marked with +. "Activate" those to expand them. "Read-only" addresses are marked with *. The values at those addresses get updated by the game on every frame, so you can't change or freeze them. You'd need to disassemble the part of the code that updates them and instruct it to write a different value.
To scan for and add addresses to the table you can follow generic CE tutorials. In JSRF, very few addresses are static and will work the next time you launch the game. Generally, after you've scanned for an address successfully, you have to pointer scan for it as well. Simply put, a pointer is an address whose value is another address. Multiple pointers are often chained together, only the first one having a static address. Search for pointer scan tutorials to see how CE can do this work for you.