cortex-command-community / **Cortex-Command-Community-Project** Public

develo… · **33** Branches · **24** Tags · Go to file · Go to file · Code

Causeless Merge pull request #157 from cortex-command-community/minor-fixes ✓

afb20b7 · 7 hours ago · 10,236 Commits

.github	macos, windows meson builds	2 weeks ago
.vscode	fix vscode default launch to no config c…	last month
Data	AddRequiredArea for Refinery Assault	12 hours ago
Documentation	fix doxyfile for new monorepo structure	9 months ago
Licences	add sdl license	2 years ago
Resources	install appstream and scalable icon wit…	last month
Source	Fixed GetArea fuckery in the laziest w…	7 hours ago
external	macos, windows meson builds	2 weeks ago
.clang-format	Back to align. For whatever reason thi…	9 months ago
.git-blame-ignore-revs	Updated git ignore revs	9 months ago
.gitignore	Add AppImage to gitignore	7 months ago
CHANGELOG.md	minorly fix the changelog...	10 hours ago
LICENSE	Initial commit	last year
LogFmod.txt	new basic explo sounds	last year
README.md	Update README.md	3 months ago
RTEA.common.props	Move msbuild defines to common.props	10 months ago
RTEA.sln	Rename Tracy->Profiling, output debu…	last year
RTEA.vcxproj	missing windows files	2 weeks ago
RTEA.vcxproj.filters	missing windows files	2 weeks ago
cpp.hint	Update cpp.hint and some minor form…	2 years ago
meson.build	macos, windows meson builds	2 weeks ago
meson_options.txt	fix macos release packaging	9 months ago
meson_osxcross.txt	continue using system lib on macos cr…	6 months ago

README · AGPL-3.0 license

Cortex Command Community Project Source

The Cortex Command Community Project is Free/Libre and Open Source under GNU AGPL v3

Meson Build (Linux, macOS) passing · Windows Build no status

This is a community-driven effort to continue the development of Cortex Command.
Stay up to date in our Discord channel.

Installing the Game

If you just want to play the latest version of the game you can get it from our website.

Getting Mods

You can get mods from our mod portal.

How To Make Your Voice Heard

About

Cortex Command - Open Source under GNU AGPL v3

cortex-command-community.github.io/

#cortex-command

Readme

AGPL-3.0 license

Activity

Custom properties

113 stars

7 watching

14 forks

Report repository

Releases 6

Release v6.2.2 Latest
on Feb 24

+ 5 releases

Packages

No packages published

Contributors 41

+ 27 contributors

Languages

● **C++** 63.5%	● **Lua** 18.3%
● **C** 17.9%	● **Meson** 0.2%
● **HTML** 0.1%	● **GLSL** 0.0%

So you want to take part in the project? A good start would be going to the discord where the project is discussed. You can find a link here. All our releases are available under the releases area and all of our bugs, changes, and ideas are tracked in GitHub issues. Feel free to play and suggest changes or point out any problems.

How To Make Issues

Please feel free to add issues and bugs. It's as simple as going to the issues tab and clicking a button. Once you do that, you'll see an easy to follow template to fill in. Try to put in the appropriate category for the issue and it'll be handled from there.

How to Contribute

If you've got any C++ experience, experience with the game's ini data through modding it, are good at spriting, or know Lua, you can contribute some of your time directly to the project. We'll gladly consider all pull requests that come in and are always happy to have more hands on deck.

More Information

See the Information and Recommendations page for more details and useful development tools.

Windows Build Instructions

First you need to download the necessary files:

1. Install the necessary tools.
   You'll probably want Visual Studio Community Edition (build supports 2019 (>=16.10) and 2022 versions. Earlier versions are not supported due to lack of C++20 standard library features and conformance).
   You also need to have both x86 and x64 versions of the Visual C++ Redistributable for Visual Studio 2015-2022 installed in order to run the compiled builds.
   You may also want to check out the list of recommended Visual Studio plugins here.

2. Clone this Repository into a folder.

3. Copy the following libraries from `Cortex-Command-Community-Project\external\lib\win` into the root directory:

- `fmod.dll`
- `SDL2.dll`

  For 32-bit builds, copy the following libraries from the `x86` folder inside `...\lib\win` as well:

- `fmodL.dll`
- `SDL2-32.dll`

Now you're ready to build and launch the game.
Simply open `RTEA.sln` with Visual Studio, choose your target platform (x86 or x64) and configuration, and run the project.

- Use `Debug Full` for debugging with all visual elements enabled (builds fast, runs very slow).
- Use `Debug Minimal` for debugging with all visual elements disabled (builds fast, runs slightly faster).
- Use `Debug Release` for a debugger-enabled release build (builds slow, runs almost as fast as Final. **Debugging may be unreliable due to compiler optimizations**).
- Use `Final` to build release executable.

The first build will take a while, but future ones should be quicker.

If you want to use an IDE other than Visual Studio, you will have to build using meson. Check the Linux and Installing Dependencies section for pointers.

Windows Subsystem for Linux (WSL)

The Linux build can be built and run on Windows 10 using WSL by following the Linux building and running instructions.
Information on installing and using WSL can be found here.

Building can be done directly from the Windows filesystem side, without having to clone the repositories on the Linux filesystem side.
By default WSL will mount your `C:` drive to `/mnt/c/`, or just `/c/`. From there you can navigate to the Source and Data directories to follow the meson build steps.

This has been tested with WSL2 Ubuntu 22.04 but should work with other distributions and WSL1 as well.

Linux and macOS Build Instructions

The Linux build uses the meson build system, and builds against system libraries.

Dependencies

- `meson` `>= 1.0.0` (`pip install meson` if your distro doesn't include a recent version)
- `ninja`
- `gcc` , `g++` (>=12, clang unsupported)
- `sdl2`
- `opengl` (usually provided by the gpu driver)
- `flac`
- `luajit`
- `lua` (maybe optional)
- `minizip`
- `tbb`
- `lz4>=1.9.0`
- `libpng`
- `dylibbundler` (required only if installing on macOS)
- `SDL2_image` (linux only)

For unspecified versions assume compatibility with the latest ubuntu LTS release.

Building

1. Install Dependencies (see below for instructions).
2. Clone this Repository and open a terminal in it.
3. `meson setup build` or `meson setup --buildtype=debug build` for debug build (default is release build)
   For macOS you need to specify gcc, with `env CC=gcc-13 CXX=g++-13 meson setup build`
4. `ninja -C build`
5. (optional) `sudo ninja install -C build` (To uninstall later, keep the build directory intact. The game can then be uninstalled by `sudo ninja uninstall -C build`)

If you want to change the buildtype afterwards, you can use `meson configure --buildtype {release or debug}` in the build directory or create a secondary build directory as in Step 3. There are also additional build options documented in the wiki as well as through running `meson configure` in the build directory.

Running

(If you installed the game in step 5 above, it should appear with your regular applications and will just run)

1. (*optional*) Copy (link) all `libfmod` files from `external/lib/[os]/[arch]` into the repository.

- Linux: `cd $REPOSITORY; ln -s ../external/lib/linux/x86_64/libfmod.so* .`
- macOS: `cd $REPOSITORY; ln -s ../external/lib/macOS/libfmod.dylib .`

2. Run `./CortexCommand` or `./CortexCommand_debug`.

Installing Dependencies

macOS additional dependencies:

- `brew` brew.sh (or any other package manager)
- `Xcode` or `Command Line Tools for Xcode` (if you need to, you can also generate an xcode project from meson using the `--backend=xcode` option on setup)

Homebrew (macOS):
`brew install pkg-config sdl2 minizip lz4 flac luajit lua libpng tbb gcc@13 ninja meson dylibbundler`

Arch Linux:
`sudo pacman -S sdl2 sdl2_image tbb flac luajit lua minizip lz4 libpng meson ninja base-devel`

Ubuntu >=22.04:
`sudo apt-get install build-essential libsdl2-dev libsdl2-image-dev libloadpng4-dev libflac++-dev luajit-5.1-dev liblua5.1-dev libminizip-dev liblz4-dev libpng++-dev libtbb-dev ninja-build python3-pip`
`sudo python3 -m pip install meson`

Fedora:
`# dnf install allegro-loadpng-devel allegro-devel libsdl2-devel SDL2_image-devel lua-devel boost-devel meson ninja-build flac-devel luajit-devel minizip-compat-devel tbb-devel lz4-devel libpng-devel lua-devel gcc gcc-c++`

Troubleshooting

- older versions of `pipewire(-alsa)` and fmod don't work well together, so the game might not close, have no sound or crash. Workaround by `ln -s /bin/true /usr/bin/pulseaudio`

Debugging with VS Code

This repository includes launch configurations to automatically build and debug the game using VS Code on any of the supported platforms using one of the two supported build systems.

Requirements

- C/C++ Extension Pack extension (all platforms)

msbuild *(Windows only)*

- msbuild command line tools (available here), available on system `PATH`
- The `fmod.dll` library must be copied to the **Data Repository** (as above)

meson *(All platforms)*

- meson, as above, available on the system `PATH`
- The meson editor extension
- Run the provided `Setup Meson` task, found via the command palette -> `Tasks: Run Task`
- Windows:
  - Visual Studio (2022) C++ Build Tools (`MSVC v143`)
  - The `fmod.dll` library must be copied to the **Data Repository** (as above)
- Linux:
  - All the dependencies listed above
- macOS:
  - All the dependencies listed above
  - The `lldb` debugger

These launch configurations are accessible via the Run and Debug view, and provide profiles to build and run the game in Release mode or any of the 3 Debug modes.