Build instructions

Notice: We have switched from Scons to CMake in release 0.9.16. Scons is no longer available as build system after release 0.10.12.

First you need a copy of the code (if you intend on working on the game, use your fork's URL here):

```
> git clone https://github.com/endless-sky/endless-sky
```

You can get a copy of the code either using "git clone," or using the repo's download button to obtain a .ZIP archive.

The game's root directory, where your unzipped/ `git clone` d files reside, will be your starting point for compiling the game. You can use `cd endless-sky` to enter the game's directory.

Next, you will need to install a couple of dependencies to build the game. There are several different ways to build Endless Sky, depending on your operating system and preference.

- Windows
- Windows (MinGW)
- MacOS
- Linux

Installing build dependencies

Windows

We recommend using the toolchain from Visual Studio to build the game (regardless of the IDE you wish to use).

Download Visual Studio (if you do not want to install Visual Studio, you can alternatively download the VS Build Tools), and make sure to install the following components:

- "Desktop Development with C++",
- "C++ Clang Compiler for Windows",
- "C++ CMake tools for Windows".

We recommend using Visual Studio 2022 or newer. If you are unsure of which edition to use, choose Visual Studio Community.

Windows (MinGW)

We recommend the MinGW Winlibs distribution, which also includes various tools you'll need to build the game as well. It is possible to use other MinGW distributions too (like Msys2 for example), but you'll need to make sure to install CMake (3.21 or later) and Ninja.

When installing MinGW, use the following settings:

- Version: `8.1.0` (or greater; this document assumes 8.1.0)
- Architecture: `x86_64`
- Threads: `posix`
- Exception: `seh`

You'll need the POSIX version of MinGW. If you want your builds to have the same runtime library requirements as the official releases of Endless Sky, choose a version that links to the UCRT. For the Winlibs distribution mentioned above, the latest version is currently gcc 14 (direct download link). Download and extract the zip file in a folder whose path doesn't contain a space (C:\ for example) and add the bin\ folder inside to your PATH (Press the Windows key and type "edit environment variables", then click on PATH and add it to the list).

Endless Sky requires precompiled libraries to compile and play: Download link

MacOS

Install Homebrew. Once it is installed, use it to install the tools and libraries you will need:

```
$ brew install cmake ninja mad libpng jpeg-turbo sdl2 minizip libavif catch2 flac
```

Note: If you are on Apple Silicon (and want to compile for ARM), make sure that you are using ARM Homebrew!

If you want to build the libraries from source instead of using Homebrew, you can pass `-DES_USE_SYSTEM_LIBRARIES=OFF` to CMake when configuring.

Linux

You can use your favorite package manager to install the needed dependencies. If you're using a slower moving distro like Ubuntu or Debian (or any derivatives thereof), make sure to use at least Ubuntu 22.04 LTS or Debian 12. If your distro does not provide up-to-date version of these libraries, you can use vcpkg to build the necessary libraries from source by passing `-DES_USE_VCPKG=ON`. Older versions of Ubuntu and Debian, for example, will need this. Additional dependencies will likely need to be installed to build the libraries from source as well.

In addition to the below dependencies, you will also need CMake 3.19 or newer, however 3.21 or newer is strongly recommended. You can get the latest version from the official website. If you are often switching branches, then you can also consider installing ccache to speed up rebuilds after switching branches.

▶ DEB-based distros

▶ RPM-based distros

Building the game

Building from the command line

(Note: This commands require CMake 3.21+. If you are using a lower version of CMake you will not be able to use the presets mentioned in this section.)

Here's a summary of every command you will need for development:

```
$ cmake --preset <preset>                                  # configure project (only needs
$ cmake --build --preset <preset>-debug                    # build Endless Sky and all test
$ cmake --build --preset <preset>-debug --target EndlessSky    # build only the game
$ ctest --preset <preset>-test                             # run the unit tests
$ ctest --preset <preset>-benchmark                        # run the benchmarks
$ ctest --preset <preset>-integration                      # run the integration tests (Lir
```

The executable will be located in `build/<preset>/Debug/`. If you'd like to debug a specific integration test (on any OS), you can do so as follows:

```
$ ctest --preset <preset>-integration-debug -R <name>
```

You can get a list of integration tests with `ctest --preset <preset>-integration-debug -N`.

(You can also use the `<preset>-release` preset for a release build, and the output will be in the Release folder).

Replace `<preset>` with one of the following presets:

- Windows: `clang-cl` (builds with Clang for Windows), `mingw` (builds with MinGW), `mingw32` (builds with x86 MinGW)
- MacOS: `macos` or `macos-arm` (builds with the default compiler, for x64 and ARM64 respectively)
- Linux: `linux` (builds with the default compiler), `linux-gles` (compiles with GLES instead of OpenGL support), `linux-armv7` (provides better support for 32-bit ARM systems)

You can list all of available presets with `cmake --list-presets`.

▶ Alternative linkers

Using an IDE

Most IDEs have CMake support, and can be used to build the game. We recommend using Visual Studio Code.

Visual Studio Code

After installing VS Code, install the C/C++ and CMake Tools extensions, and open the project folder under File -> Open Folder.

▶ Other recommended extensions

You'll be asked to select a preset. Select the one you want (see the list above in the previous section for help). If you get asked to configure the project, click on Yes. You can use the bar at the very bottom to select between different configurations (Debug/Release), build, start the game, and execute the unit tests. On the left you can click on the test icon to run individual integration tests.

Visual Studio

We recommend using Visual Studio 2022 or newer, because of its better CMake integration. Once you have installed Visual Studio, you can simply open the root folder.

▶ Step-by-step instructions for Visual Studio 2022 or later

If you are on an earlier version of Visual Studio, or would like to use an actual VS solution, you will need to generate the solution manually as follows:

```
> cmake --preset clang-cl -G"Visual Studio 17 2022"
```

This will create a Visual Studio 2022 solution. If you are using an older version of VS, you will need to adjust the version. Now you will find a complete solution in the `build/` folder. Find the solution and open it and you're good to go!

Code::Blocks

If you want to use the Code::Blocks IDE, from the root of the project folder execute:

```
> cmake -G"CodeBlocks - Ninja" --preset <preset>
```

With `<preset>` being one of the available presets (see above for a list). For Windows for example you'd want `clang-cl` or `mingw`. Now there will be a Code::Blocks project inside `build\`.

XCode

If you want to use the XCode IDE, from the root of the project folder execute:

```
$ cmake -G Xcode --preset macos # macos-arm for Apple Silicon
```

The XCode project is located in the `build/` directory.