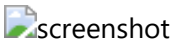
🚀 Space-Something (working title)

Arcade-style top-down space shooter written in **C++17** and **raylib 5**.
Runs on Windows, Linux and macOS.



---

1 · Features

- Pixel-art ship with modular sprite-parts (thrusters, weapons, …)
- Player-controller wrapper that can swap to bigger ships later
- Camera that follows any `CameraTarget` entity
- Component-based world grid for lightweight collision / culling
- Pure CMake build – no Makefile hacks – ships with raylib sources

---

2 · Prerequisites

	Windows	**Linux / macOS**
Compiler	MinGW-w64 10+ (auto-downloaded with VS Code C/C++)	GCC 10+ / Clang 12+
CMake	≥ 3.20	`sudo apt install cmake` or `brew install cmake`
Ninja	*(optional)* `choco install ninja`	`sudo apt install ninja-build` or `brew install ninja`

> **No global raylib install is required** – the build pulls the exact tag we need.

---

3 · Getting the code

```
git clone --recursive https://github.com/your-nick/space-something.git
cd space-something
```

Using `--recursive` is only needed if you keep raylib as a git-submodule.
With FetchContent (default), a plain `git clone` is enough.

---

4 · Building (Debug)

```
# 1. Generate a Ninja build folder in ./build
cmake -S . -B build -G Ninja -DCMAKE_BUILD_TYPE=Debug
```

```
#   ↳ First configure will automatically download raylib 5.0.0,
#     configure it as a sub-project and write Ninja files.

# 2. Compile both raylib and the game
cmake --build build

# 3. Run
./build/bin/game          # or game.exe on Windows
```

Release build

```
cmake -S . -B build/release -G Ninja -DCMAKE_BUILD_TYPE=Release
cmake --build build/release
```

---

5 · Project layout

```
.
├─ CMakeLists.txt          ← root build script
├─ external/               ← *optional* git-submodule with raylib
├─ include/                ← public headers
├─ src/                    ← *.cpp
├─ rsc/                    ← textures / audio / levels
├─ bin/                    ← (auto) final executables land here
└─ obj/                    ← (auto) object files
```

How CMake fetches raylib

We use `FetchContent` (CMake's built-in package downloader):

```
include(FetchContent)
FetchContent_Declare(
  raylib
  GIT_REPOSITORY https://github.com/raysan5/raylib.git
  GIT_TAG        5.0.0
)
FetchContent_MakeAvailable(raylib)
```

- The very first `cmake -S . -B build` clones that exact commit into `build/_deps/raylib-src`.
- Afterwards it is treated like any other sub-directory target –
  no system install or PATH fiddling required.

> **Prefer a submodule instead?**
> Just delete the `FetchContent_…` block and add
> `add_subdirectory(external/raylib)` – the rest of this README still works.

---

6 · Packaging a release build

```
cmake --install build/release --prefix dist          # copies game + assets
cp -r rsc dist/                                      # copy resources
cd dist && zip -r SpaceSomething-1.0-windows.zip *   # or tar.gz on Linux
```

Upload the resulting archive to **GitHub → Releases**, itch.io, Steam…
Keep binaries *out* of git history.

---

7 · Troubleshooting

Problem	Fix
"No CMAKE_CXX_COMPILER found"	Install/point VS Code C/C++ extension to MinGW-w64
raylib include errors	Make sure you did **not** install an old raylib in Program Files that might override headers in PATH
Link errors about `winmm`/`pthread`	Delete `build/`, rerun `cmake -S . -B build` – changing compilers requires a clean configure

---

8 · Licence

MIT © 2025 Your Name
raylib is zlib/libpng © Ramon Santamaria – see `external/raylib/LICENSE`.

```
---

### Why/How the build “pulls” raylib

* **`FetchContent`**: CMake contacts GitHub, clones the desired tag and adds it as an ordinary sub-directory. Users do **not** need to pre-install anything.
* **Optionally** you can keep raylib as a **git submodule**. Contributors clone with `--recursive` (or run `git submodule update --init` once). Still no system install needed.

Either approach keeps **all dependencies self-contained** inside the repo – the user only installs toolchain + CMake once.
```

```
Feel free to tweak paths (`external`, `bin/`, `obj/`) or add presets, but the
README above should give anyone a 5-minute path from *clone* → *running game*.
```