Guide contents

Choose a modding path

Set up a small WolvenKit project

Add behavior only when the asset path works

Package, test and diagnose

Sources / References

Documentation: In progress **Quick start: In progress**

Offline copy from ModDocs. Source links remain clickable.

Cyberpunk 2077 modding

Cyberpunk 2077 has several distinct modding routes. **REDmod** is CD Projekt RED's official modding toolset; **WolvenKit** is a community editor that can manage projects and integrate with REDmod. Gameplay scripting and engine extensions use separate community frameworks. CD Projekt RED's modding support page explains the official tool, while the Red Modding wiki documents community workflows.

> **Quick start:** make one small asset or data change in a WolvenKit project, install it into a test profile, and verify that the game loads it. Keep your game version and each framework version together in your project notes.

Choose a modding path

Goal	Start with	Watch for
Replace or add a game asset	WolvenKit project guide	Paths, dependencies and archive priority
Work with official REDmod tooling	CDPR modding support and REDmod documentation	Matching the tool and game builds
Change scripted behavior	Red Modding guides	Runtime frameworks and version compatibility

Goal	Start with	Watch for
Edit a world or quest	Topic-specific Red Modding guides	Many linked resources, states and dependencies

Do not install every framework for a simple asset replacement. Select the path your project needs and record each dependency you add. For a first test, use a separate mod profile with unrelated mods disabled. This makes a broken resource path much easier to diagnose.

Set up a small WolvenKit project

Install a WolvenKit release suitable for the current game version. Create a project with a unique name and keep source files separate from packaged output. The WolvenKit project documentation describes its project layout and asset browser. The first-mod walkthrough covers a basic edit and installation.

Choose a visible, reversible change. Import or override only the resource required for that change. Check the exact resource path and whether another file references it. Build or install the project using the documented WolvenKit action, launch the game, and inspect the target in a save where it appears. Then remove the test mod and confirm the base behavior returns. That last check helps identify whether your project actually caused the result.

Add behavior only when the asset path works

If you need a new interaction or gameplay rule, select the framework documented for that feature rather than assuming an archive can do it alone. The Red Modding getting-started index separates mod types. When an example requires another mod, extension or script runtime, record that dependency and its version. Test the minimal behavior before adding UI, localization or multiple assets.

Package, test and diagnose

Package only your own work and required metadata, preserving the paths expected by the selected framework. Do not bundle the game's original archives. Test from the packaged copy, not from loose files left by the editor. If the game launches but the change is absent, inspect installation path, mod activation and resource names. If it fails at startup, disable the test mod and its dependency chain, then re-enable components one by one. Check the current Red Modding troubleshooting guide before using an old fix written for another patch.

This guide is **unfinished**. It covers the main development choices and a small project lifecycle but not a tested custom quest, item, character, animation or script tutorial. The linked official and community references provide version-specific details for each branch.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- modding support page
- Red Modding wiki
- WolvenKit project guide
- REDmod documentation
- Red Modding guides
- WolvenKit project documentation
- Red Modding troubleshooting guide