Guide contents

Choose how to build it

Prepare the development environment

Make a first mod

Test in the game

Package and release

A complete first project: recipe and unlock

Sources / References

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

Offline copy from ModDocs. Source links remain clickable.

Satisfactory modding

Satisfactory mods commonly use the community's Satisfactory Mod Loader (SML), Unreal project setup, and Alpakit for packaging. There are Blueprint and C++ development paths; ContentLib offers a separate data-oriented route for some content. The maintained Satisfactory Modding Documentation is the version-specific source.

> **Quick start:** read the current Getting Started instructions, set up the provided starter project and required editor, make a uniquely named mod reference, then follow the Simple Mod path. Build and test one small change before adding models or code.

Choose how to build it

Goal	Route	What to learn first
New item, recipe, or buildable with editor support	Unreal project and Blueprint	Starter project, plugin setup, content registration, Alpakit

Goal	Route	What to learn first
New systems or complex behavior	C++ plus Unreal project	C++ module setup, SML APIs, build targets
Data-driven content without a full Unreal content workflow	ContentLib	Its own setup, schema, and supported content types

The official community docs advise new developers to start with Blueprint before C++. ContentLib setup is useful when its supported data format fits the mod, but it is not a substitute for every Unreal asset or behavior.

Prepare the development environment

Use the current SML starter project and the Unreal Editor build specified by the docs for your Satisfactory version. Engine and tool versions matter: a random stock Unreal installation may not match the game. Follow project setup exactly for the required downloads, build configuration, and game path. The docs distinguish the `Development Editor` build used to work in the editor from shipping output that Alpakit packages.

Create a **mod reference**, a stable machine-readable name used across plugin, folder, and class names. It cannot contain spaces or punctuation such as `+` or `.`. Keep it unique and use the same spelling throughout the project. The Getting Started guide explains the naming rules and why a mismatch causes later problems.

Make a first mod

In the starter project, create your plugin through the documented Alpakit workflow. Learn how its plugin setup and Game World Module register content; then follow one piece of the Simple Mod tutorial, such as a recipe and its unlock. A content asset must be both defined and reachable in game. Test the unlock path, crafting result, display text, and icon separately. After that works, move to a buildable, machine, or more elaborate system.

For C++ work, use the current C++ setup rather than guessing modules or linking against game binaries by name. Build the editor target after adding a module; then

package a shipping build through Alpakit. A successful C++ compile does not prove the plugin will load in game.

Test in the game

Build with Alpakit, launch the game with SML, and check the loader and game logs for registration or dependency errors. Start with just the new mod and its required dependencies. Verify the content in a new save, then check an existing save if the mod changes progression. If the feature is intended for multiplayer, test both host and client; the release guide explicitly calls out multiplayer testing.

When something fails, narrow the stage: editor/build failure, packaging failure, loader failure, asset registration failure, or gameplay failure. A mod that packages successfully may still lack an unlock, missing asset, or runtime dependency. Do not change several stages at once while diagnosing it.

Package and release

The plugin's `.uplugin` file holds key metadata such as version, dependencies, and credits. The release guide recommends editing it through Alpakit, repackaging, and testing the package you actually plan to upload. Publish through the Satisfactory Mod Repository only after the packaged version works with the target game and SML versions.

A complete first project: recipe and unlock

The maintained Simple Mod course begins with a recipe that turns leaves into wood, then adds a HUB milestone to unlock it. That pair is a better first deliverable than an unreachable recipe asset. Set up the current SML starter project and game-matched Unreal Editor first. Create a uniquely named plugin, add the required Game World Module registration, then follow the course's recipe and milestone steps in order. Save and build after each stage so a missing module or asset reference is caught close to the edit that caused it.

In game, check that the mod loads, the milestone appears, purchasing it makes the recipe available, and the input/output quantities are what the asset specifies. Test in a fresh save; progression state in an older save can conceal an unlock error. Package through Alpakit, remove the unpackaged working copy, and repeat the milestone-to-crafting path with the packaged build. If the recipe exists in an editor view but not in gameplay, inspect the registration module and unlock chain before changing its icon or mesh. If Alpakit fails, resolve that build stage before debugging game content.

This is a complete small Blueprint/content route. The later items, buildables, machines, and C++ sections in the course are separate extensions, each with additional assets and testing requirements.

Still being expanded: this page is a detailed route through the toolchain, but not yet a tested custom-recipe project or API-by-API reference. The maintained source contains the exact downloads, game-version requirements, and stepwise Unreal screenshots, which can change between releases.

Sources / References

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

- Satisfactory Modding Documentation
- Getting Started
- Simple Mod
- ContentLib setup
- project setup
- C++ setup
- release guide