

Dein PS2-Spiel. Dein **Projekt.**

Elf Skills für einen eigenen PC-Port und kleine Mods. Hier findest du den Einstieg und kopierbare Aufträge.

Das Paket arbeitet mit eigenen PS2-Spielen. Ein fertiger Port ist kein Sofort-Ergebnis: Analyse und Fehlersuche können Tage oder Wochen dauern. Manche Spiele bleiben unspielbar.

01 Material bereitlegen

Eigene PS2-Disc, Computer mit DVD-Laufwerk oder bereits selbst erstelltes Abbild. Dazu [Claude Code nach offizieller Anleitung installieren und anmelden](#), [Git](#) und [Python ab 3.8](#).

02 Im leeren Projektordner starten

Ordner Konsole-Test anlegen und dort ein Terminal öffnen.
Windows: im Explorer in die Adresszeile powershell tippen.
Danach `claude` starten und anmelden.

03 In Claude Code einzeln senden

```
/plugin marketplace add Jakeschincariol/console-skill
```

```
/plugin install console-skill@console-skill
```

Installationsmeldung lesen. Falls verlangt neu laden. Danach / tippen und `console-reverse` suchen.

Skill: MIT, kostenlos. Claude-Nutzung separat. Für den eigentlichen Port brauchst du [PS2Recomp](#), CMake ab 3.20 und C++20-Compiler. Der Startauftrag auf Seite 2 klärt deinen konkreten Stand. [Originalpaket öffnen](#) ↗

01 / DER ERSTE NACHVOLLZIEHBARE SCHRITT

Erst die Disc verstehen. Dann den Port aufbauen.

Ersetze die drei eckigen Felder und sende den Auftrag im selben Claude-Code-Projekt.

STARTAUFRAG

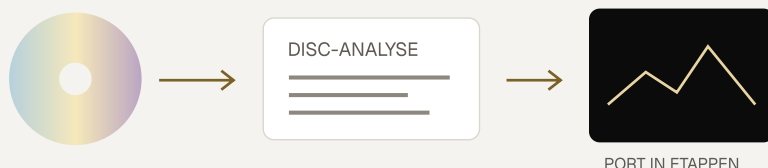
```
/console-skill:console-reverse
```

Du begleitest meinen PS2-Port. System: [Windows, macOS oder Linux]. Spiel: [Titel und Region]. Material: [eigene Disc oder Pfad des eigenen Abbilds].

Lies oder erstelle `console/STATE.md`. Prüfe vorhandene Programme und echte Pfade. Fehlt PS2Recomp, CMake, der C++20-Compiler oder Python, führe mich mit offiziellen Links durch die fehlenden Schritte.

Danach: Disc prüfen, Boot-Datei, Seriennummer und Region ermitteln. Speichere den belegten Stand und den nächsten Schritt in `console/STATE.md`. Keine Spieldownloads, nichts überschreiben, keinen Build löschen. Vor einem großen Build Umfang erklären.

Prüfe Werkzeugausgaben, korrigiere Fehler und prüfe erneut. Trenne durchgeführte von offenen Tests. Prüfe aus simulierter Nutzersicht, ob Ergebnis und nächste Aktion verständlich sind.



Das ist dein erstes Ergebnis.

Eine Projektdatei mit echten Pfaden, erkannter Spielversion und offenen Punkten. Noch kein garantiert spielbares Spiel.

Für handelsübliche Spiele empfiehlt das Paket [Ghidra](#) mit [EmotionEngine Reloaded](#). Den ausführlichen [Startauftrag online kopieren ↗](#). Die Grafik erklärt den Ablauf.



02 / WELCHER SKILL MACHT WAS?

Elf Aufgaben. Ein gemeinsamer Stand.

Alle lesen `console/STATE.md`. Als Plugin steht vor jedem Befehl `/console-skill:.`

reverse



backend ↔ tester



dein Mod

console-reverse

Disc, Programmdatei und Projektstand erfassen. Hier starten.

console-backend

Fehlende Systemfunktionen, Dateizugriff und Boot-Probleme bearbeiten.

console-tester

Port starten, Abstürze auswerten und Änderungen erneut testen.

console-graphics

Texturen und Bildausgabe untersuchen, eigene Grafiken einsetzen.

console-controller

Knöpfe, Sticks und Tastatur passend zuordnen.

console-mod

Einen konkreten Wunsch als Patch oder Override umsetzen.

console-levels

Dateistruktur und Archive untersuchen, eigene Änderungen einbauen.

console-sound

Musik und Effekte untersuchen, eigene Audiodateien einsetzen.

console-physics

Zum Beispiel Geschwindigkeit, Schwerkraft oder Sprunghöhe finden.

console-saves

Eigene Speicherstände kopieren und Konflikte nachvollziehbar halten.

console-anywhere

Zielgerät prüfen und ein Paket ohne fremde Spieldaten vorbereiten.

Zuerst Boot und Spielbarkeit erreichen, dann Änderungen angehen. Vollständiger Beispielaufruf: `/console-skill:console-controller`. [Originaldokumentation ↗](#)

03 / DER ERSTE KLEINE MOD

Ein Auto schneller. Den Rest erhalten.

Beispielauftrag für ein eigenes Rennspiel, dessen Port bereits spielbar ist. Tausche das Ziel gegen deinen konkreten Wunsch.

VORHER

Dieselbe Strecke.
Dasselbe Spielerauto.
Originalgeschwindigkeit.

GEWÜNSCHTE ÄNDERUNG

Spielerauto etwa 20 % schneller.
Gegner, Kamera und Spielstände
bleiben gleich.

MOD-AUFTRAG

`/console-skill:console-mod`

Lies den Projektstand. Finde mit `console-physics` den belegten Geschwindigkeitswert für mein Spielerauto. Ziel: etwa 20 % schneller, ohne Gegner oder Steuerung zu ändern.

Erkläre Fundstelle und Rücknahme. Sichere Konfigurationen. Erstelle einen kleinen Patch oder Override in `console/patches/`; Originale, generierten Code und Header nicht ändern. Inkrementell bauen und mit `console-tester` kurz starten.

Vergleiche dieselbe Strecke vor und nach der Änderung. Lass mich Fahren, Bremsen, Lenken und Speichern ausprobieren. Fehler zurücknehmen, korrigieren und erneut prüfen. Nach drei Fehlschlägen am selben Problem Befund und konkrete Rückfrage liefern. Echte Tests von offenen Punkten trennen und Verständlichkeit aus simulierter Nutzersicht prüfen.

Controller anpassen: `/console-skill:console-controller` aufrufen, Modell und Anschluss nennen. Jeden Knopf und beide Sticks im Spiel selbst prüfen. Vibration ist derzeit nicht angeschlossen.

[Mod- und Controller-Auftrag online kopieren ↗](#)

[Trag dich gern in die Community-Warteliste ein. ↗](#)

Stand: 23.10.2026. Quelltexte und Befehle geprüft; kein Spiel portiert, kein Claude-Lauf und kein Controller-Test ausgeführt. Eigene Disc- und Spieldateien bleiben lokal. Teile nur eigene Patches und Inhalte.