

# Ziele der Laborübung

- Grundverständnis: Was sind Blender und UV? Wie funktioniert MCP?
- Wie man über Prompts einfache 3D-Objekte erstellt und verändert.
- Automatische Generierung eines Bilds in Blender, z. B.:



## Erklärung der benutzten Technologien

### Blender

Blender ist ein freies, quelloffenes 3D-Produktionswerkzeug, mit dem sich Animationen, Modelle, Simulationen und visuelle Effekte erstellen lassen. Die Vorteile von Blender gegenüber proprietären Lösungen liegen vor allem in der offenen Erweiterbarkeit, der aktiven Community sowie der breiten Plattformunterstützung (Linux, Windows, macOS).

Die Software integriert zahlreiche Funktionen – von Modellierung, Texturierung und Rigging über Compositing bis hin zu Video-Editing – und besitzt eine leistungsfähige Python-Schnittstelle, über die externe Systeme (wie KI-Modelle) direkt mit Blender interagieren können.

### MCP (Model Context Protocol)

Das Model Context Protocol ist ein offener Standard, der KI-Modellen eine einheitliche, sichere Kommunikation mit externen Datenquellen, Tools und Anwendungen ermöglicht. MCP ist dabei transportagnostisch, d. h. es kann über beliebige Kommunikationskanäle betrieben werden, solange Nachrichten strikt im JSON-RPC-Format übertragen und die Lebenszyklusregeln des Protokolls eingehalten werden.

Eine MCP-Anwendung besteht grundsätzlich aus zwei Komponenten: einem **MCP-Server**, der

Funktionen oder Ressourcen bereitstellt (in unserem Fall: Blender), und einem **MCP-Client**, der eine Verbindung herstellt, die verfügbaren Fähigkeiten ermittelt und die angebotenen Werkzeuge nutzt, um Aktionen auszuführen oder Antworten zu generieren.

## UV

[uv](#) ist ein Python-Projekt- und Paketmanagement-Tool von Astral. Es kann Dinge wie pip (installiert Python-Pakete) und venv (erstellt eine isolierte Python-Umgebung) und ist auch noch deutlich schneller.

Bei der Installation von uv werden zusätzlich zum Haupt-Tool noch zwei weitere Tools installiert:

- `uvx.exe`: startet Python-basierte Tools direkt ohne manuelle Installation
- `uvw.exe`: ist ein weiteres Zusatztool, welches allerdings nur selten gebraucht wird (zB für WebAssembly)

Um in Claude Python-basierte Erweiterungen auszuführen, wie zum Beispiel MCP-Server zu starten, benötigen wir uv beziehungsweise uvx.

## Kommunikation zwischen MCP-Client und Blender-MCP-Server

Der Austausch zwischen Client und Blender erfolgt über einen bidirektionalen Kommunikationskanal – typischerweise **stdio** (Subprozess mit stdin/stdout), **Streamable HTTP** oder wie in unserem Fall **TCP-Socket**.

Die grundlegenden Schritte sind wie folgt:

### 1. Initialisierung

Blender wird gestartet, das MCP-Addon aktiviert, und der über das Addon integrierte MCP-Server geht in den Bereitschaftsmodus, um eingehende Verbindungen anzunehmen.

### 2. Verbindungsaufbau

Der MCP-Client verbindet sich über den gewählten Transport (stdio oder HTTP-Stream) mit dem Server.

### 3. Discovery (Fähigkeitenabfrage)

Der Client ermittelt, welche Ressourcen und Tools das Blender-MCP-Addon bereitstellt – etwa `create_primitive`, `set_object_property` oder `execute_blender_code`.

### 4. Ausführung von Befehlen

Der Client sendet JSON-RPC-Requests an den Server; das Addon führt die gewünschten Aktionen innerhalb von Blender aus.

### 5. Antwortübermittlung

Der Server liefert standardisierte JSON-RPC-Antworten zurück, beispielsweise Objekt-IDs oder Statusmeldungen.

### 6. Folgeaktionen

Basierend auf den Ergebnissen kann der Client weitere Operationen durchführen, etwa Materialien ändern, Szeneneinstellungen anpassen oder komplexe Automationen

orchestrieren.

# Installation der benötigten Ressourcen

## Vorbereitende Arbeiten

### Installation von Blender

Auf dem Schulrechner ist Blender bereits installiert.

Sollten Sie die Übung auf Ihrem privaten Gerät durchführen und Blender noch nicht installiert haben, können sie dies über die Blender [Homepage](#) tun.

### Installation von uv

Öffnen Sie ein PowerShell-Fenster und führen Sie diesen Befehl aus:

```
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
```

`irm`...Invoke-RestMethod (lädt ein Skript aus dem Internet herunter)

`iex`...Invoke-Expression (führt den geladenen Code sofort aus)

**Sollten Sie auf Ihrem privaten PC arbeiten, kommt oft nach diesem Befehl ein Fehler wie dieser:**

Error: PowerShell requires an execution policy in [Unrestricted, RemoteSigned, Bypass] to run uv. For example, to set the execution policy to 'RemoteSigned' please run:

```
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
```

Dieser Fehler bedeutet, dass die PowerShell das Ausführen von Skripten blockiert. Das können Sie umgehen, indem Sie die Ausführungsrichtlinien für Ihren aktuellen User ändern:

```
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
```

Wenn Sie daraufhin gefragt werden, ob Sie die Ausführungsrichtlinie ändern wollen, antworten Sie mit J.

**Vergessen Sie nicht die Richtlinien am Ende der Übung wieder zu ändern!**

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Undefined
```

### uv zum PATH hinzufügen

Damit uv und uvx überall gestartet werden kann, muss der Ordner zum PATH hinzugefügt werden. Dafür gibt es zwei Optionen:

1. Die PowerShell neustarten – die Änderungen werden dann automatisch aktiv.
2. Den Pfad direkt über die PowerShell setzen (Ersetzen Sie DEIN\_USER mit ihrem Benutzerkonto):

```
$env:Path = "C:\Users\DEIN_USER\.local\bin;$env:Path"
```

**Für diese Laborübung verwenden Sie bitte die zweite Option.**

# Installation und Konfiguration von Claude Desktop

## Claude Desktop Installation

Installieren Sie Claude Desktop über die [Homepage](#).

Nach der fertigen Installation öffnen Sie Claude und bearbeiten Sie die Datei: `claude_desktop_config.json`.

Diese finden Sie unter: Claude → Settings → Developer → Edit Config

## Bearbeitung der Konfigurationsdatei

Danach navigieren Sie zu Claude → Settings → Developer → Edit Config `claude_desktop_config.json` und fügen Sie folgende Zeilen hinzu:

```
{
  "mcpServers": {
    "blender": {
      "command": "C:\\\\Users\\DEIN_USER\\.local\\bin\\uvx.exe",
      "args": ["blender-mcp"]
    }
  }
}
```

(Ersetzen Sie DEIN\_USER mit ihrem Benutzerkonto)

Mit dieser Konfiguration wird festgelegt, dass Claude `uvx.exe` mit dem Tool `blender-mcp` starten muss, um mit Blender zu kommunizieren.

## Installation des Blender-MCP Add-ons

Damit Blender mit Claude kommunizieren kann, braucht Blender selbst auch ein Plug-In.

1. Add-on herunterladen Laden Sie `addon.py` über diesen Link herunter:  
<https://github.com/ahujasid/blender-mcp/blob/main/addon.py>
2. Öffnen Sie Blender und Navigieren Sie zu Add-ons (Edit → Preferences → Add-ons)
3. Klicken Sie auf “Install...” und wählen Sie `addon.py` aus.
4. Aktivieren Sie das Add-on indem Sie die Checkbox neben “Interface: Blender MCP” auswählen.

## Verbindung mit dem Server herstellen

Öffnen Sie Blender und das MCP-Panel, indem Sie rechts oben im Blender-Fenster auf den kleinen Pfeil klicken.

Dort befindet sich der Button „**Connect to MCP server**“. Klicken Sie auf diesen → Blender verbindet sich mit Claude.

# Praxisphase

## Ersten Tests

Öffnen Sie in Claude Desktop den Bereich „Search and Tools“ und stellen Sie sicher, dass alle

Funktionen aktiviert sind.

Überlegen Sie sich einen kreativen Prompt und senden Sie ihn ab!

Verwenden Sie dann Prompt-Verkettung: Geben Sie den Prompt an eine andere KI weiter, mit der Aufgabe, einen guten Prompt für Claude Desktop zu erstellen, und vergleichen Sie, wie erfolgreich die beiden 3D-Modelle geworden sind.

Dokumentieren Sie die Ausgaben mittels Screenshots.

## **Reflexion**

1. Welche Arbeiten könnte Ihnen MCP in Zukunft abnehmen?
2. Würden Sie MCP in Zukunft selbst verwenden, wieso bzw. wieso nicht?

---

*Viel Erfolg!* 😊