# Richtlinie zum „Software-Sprint“

# ipyblender – Wir erweitern Infrastruktur für 3D-Visualisierungen in Python.

## *Schlussbericht*

Zuwendungsempfänger:

Jan-Hendrik Müller

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS24S10 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

## Kurze Darstellung der Aufgabenstellung und Motivation

Visualisierungstools im Python Ökosystem werden täglich von Millionen von Menschen genutzt. Mein Ziel ist es, die Infrastruktur für interaktive Visualisierungen zu pflegen, zu dokumentieren und zu erweitern.

Dabei suche ich den aktiven Austausch mit der Community in Internetforen, um gemeinsam Herausforderungen zu identifizieren und passende Lösungen für das Ökosystem zu entwickeln.

Ein wichtiger Bestandteil meines Projekts ist das Python-Paket *anywidget*. Es ermöglicht erstmals, moderne JavaScript-Visualisierungstools einfach im Python-Ökosystem zu nutzen. Mit *anywidget* lassen sich interaktive Elemente wie Buttons, Schieberegler, 3D-Viewer oder Whiteboards direkt in Python-Notebooks integrieren. So können neue Workflows erstellt werden, die NutzerInnen ihre tägliche Arbeit durch einen höheren Automatisierungsgrad erleichtern.

Eine besondere Herausforderung besteht darin, den Python-Prozess mit den Interaktionen im Frontend zu synchronisieren. Dafür habe ich verschiedene Ansätze ausprobiert und gemeinsam mit der Python-Community einige bewährte Praktiken entwickelt. Diese bewährten Praktiken habe ich dokumentiert und online in Form von Forenbeiträgen und Erklärungsvideos auf YouTube zur Verfügung gestellt.

Ein weiterer Aspekt des Projekts ist es, die 3D-Grafiksoftware Blender aus Python-Notebooks heraus nutzbar zu machen.

## Beitrag von ipyblender zu den Zielen der Förderinitative „Software-Sprint“

ipyblender und die dazugehörige Infrastruktur hat das Ziel, in Forschung, Lehre und Wissenschaftskommunikation zum Einsatz kommen. In meiner Zielgruppe befinden sich unter anderem NeurowissenschaftlerInnen, die ihre medizinischen Inhalte darstellen wollen, ProgrammiererInnen, die ihre Berechnungen visualisieren möchten, aber auch Science-YouTuberInnen und DatenjournalistInnen, die datenbasierte Infografiken erstellen wollen.

In Bezug zu den Zielen des „Software-Sprints“ trägt ipyblender zur nachhaltigen Stärkung der Open-Source-Software-Infrastruktur bei. Ein zentraler Aspekt des Projekts ist die Automatisierung des Datenaustauschs zwischen verschiedenen Open-Source-Komponenten. Dadurch können EntwicklerInnen interaktive Visualisierungstools effizienter in ihre Arbeitsabläufe integrieren und die Erstellung datenbasierter Inhalte in Python und Blender vereinfachen.

## Ausführliche Darstellung der Ergebnisse

Meine Ergebnisse lassen sich in drei Meilensteine gliedern: **„Anywidget Infrastruktur“**, **„Anywidget Anwendung“** und **„Ipyblender-Visualisierungen“**.

**1. Anywidget Infrastruktur**

- Während meiner Arbeit mit Anywidget habe ich unterschiedliche Ansätze getestet, um eine Synchronisation zwischen einem Python-Notebook und JavaScript-Frontend-Paketen herzustellen. Ein „Widget“ ist hierbei ein JavaScript Frontend, welches in einem Notebook ausgeführt wird. Der erste Ansatz besteht darin, JavaScript-Code direkt aus dem Notebook Zeile für Zeile an den Browser zu senden. Diese Methode ist einfach zu implementieren und leicht verständlich, kann jedoch zu Fehlern führen. Beispielsweise treten Probleme auf, wenn bereits ein aktiver JavaScript-Prozess läuft, wodurch die gesendeten JS-Zeilen wirkungslos bleiben. Ein weiteres Problem entsteht, wenn die JS-Zeilen gesendet werden, bevor das JavaScript-Frontend vollständig geladen ist.

- Der zweite Ansatz behebt diese Schwierigkeiten mithilfe sogenannter “Traitlets”. Traitlets sind Python-Variablen, die von JavaScript aus abgerufen werden können. In diesem Fall werden die Berechnungsergebnisse zunächst in Traitlets gespeichert und können anschließend von JavaScript jederzeit abgerufen werden.

- Neben dem weit verbreiteten Framework „JavaScript“ gibt es auch spezialisiertere Frontend-Pakete wie Vue, Svelte oder React. Um React Frontend-Pakete in Python-Notebooks zugänglich zu machen, war ich an der Entwicklung eines Quickstart-CLI-Tools beteiligt, welches die Projektstruktur für ein React-Widget in einem Notebook aufsetzt. https://github.com/manzt/anywidget/pull/414

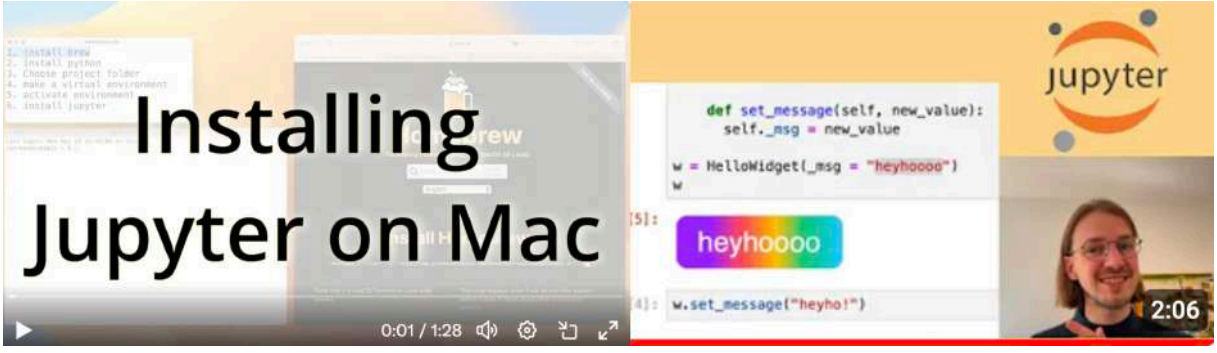



**Abb.1:** Tutorials zu Python notebooks und anywidget.

- Zudem habe ich auf YouTube Tutorials veröffentlicht (Abb.1), die den Entwicklungsprozess erklären und es neuen Nutzern erleichtern soll, eigene Widgets zu erstellen:
    - "How to Set Up Jupyter on a Mac“ https://x.com/kolibril13/status/1779887288266133634
    - "Anywidget Tutorial - Setup and first button widget in 2 minutes“ https://www.youtube.com/watch?v=eHUg2Yk-NqY
    - "How to Build a 3D Array Viewer with Python and Anywidget“ https://www.youtube.com/watch?v=RkCla8f_64w

**2. Anwendung von Anywidget: Interaktives Whiteboard und 3D-Viewer für medizinische Datensätze**

- Ein weit verbreitetes Frontend-Framework ist das interaktive Whiteboard “Tldraw”. Dieses habe ich an das Notebook-Ökosystem angeschlossen, sodass Whiteboard-Interaktionen mit dem Python-Kernel synchronisiert werden und viele spannende Anwendungsfälle möglich werden.
- **Anwendungsfall 1:** Ich habe die ChatGPT Vision-API an das Whiteboard angebunden. Hierbei werden alle Aktionen auf dem Whiteboard an OpenAI gesendet, woraufhin Python-Code zurückgegeben wird, der die Skizze in einen Plot umwandelt (Abb. 2, Links). https://x.com/kolibril13/status/1790355757445059012
- **Anwendungsfall 2:** Segmentierungsalgorithmen: Hier habe ich einen Bilderkennungs-algorithmus angeschlossen, um Elemente in einem Bild auszuwählen. In Abb. 2 (Mitte) ist zu sehen, wie Striche auf ein Bild gezeichnet werden, und die zugrunde liegenden Bildelemente (in diesem Fall historische Münzen) mit einem farbigen Label versehen werden. https://github.com/kolibril13/jupyter-tldraw
- **Anwendungsfall 3:** Reaktive Notebooks. In Python-Notebooks kommt es häufig vor, dass Zellen in ungeordneter Reihenfolge ausgeführt werden, was bei der Verwendung von ipywidgets zu Verwirrung führen kann. Dieses Problem lässt sich durch das Definieren von Callback-Funktionen beheben, allerdings ist das relativ umständlich. Deshalb habe ich nach

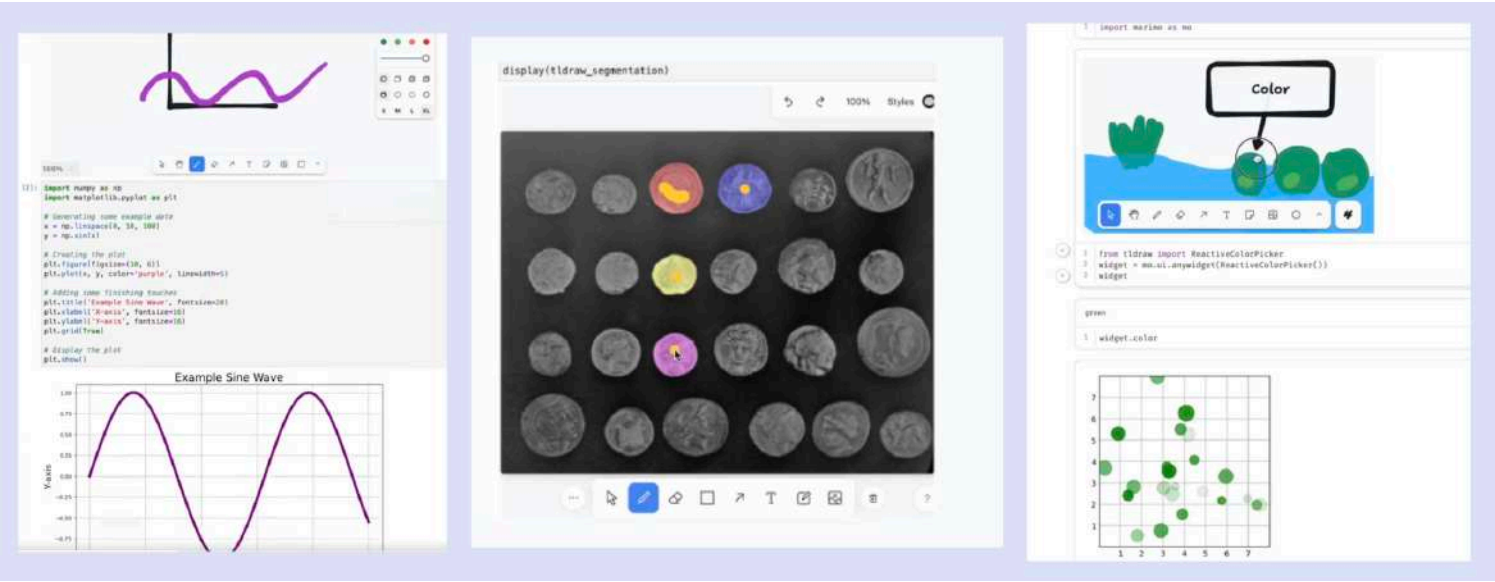



**Abb.2:** Interaktives Whiteboard in Jupyter.

Alternativen gesucht und bin dabei auf „Marimo“ gestoßen, ein Projekt, das reaktive Notebooks ermöglicht und ähnlich wie React für eine Python Umgebung funktioniert. In Abb. 2 ist rechts ein Experiment mit meinem Whiteboard-Widget zu sehen, bei dem die Interaktion im Whiteboard das Ausführen einer Python-Funktion auslöst. https://x.com/kolibril13/status/1821120339004285150

- Ich habe gemeinsam mit der SciViz-Community an der Integration des JavaScript-Viewers ‘Niivue’ in Python-Notebooks mitgewirkt. Dabei werden dreidimensionale Bilddaten und Oberflächeninformationen, wie beispielsweise ein Gehirnscan mit einem segmentierten Gewebeabschnitt, in Python geladen und interaktiv im Notebook dargestellt (Abb.3). https://github.com/niivue/ipyniivue

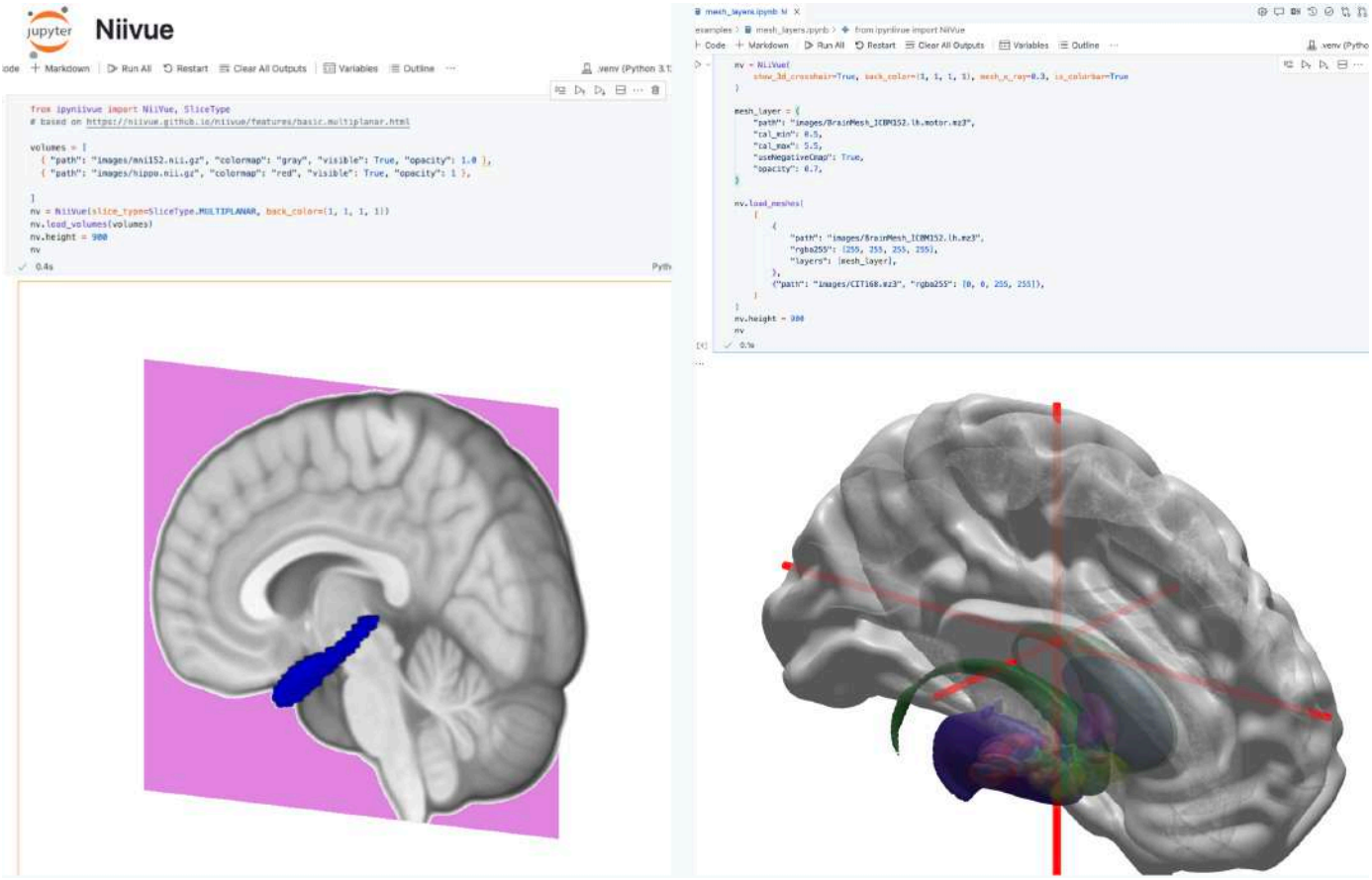



**Abb. 3:** Zwei Darstellungen von Gehirndaten in einem Python-Notebook.

**3. Ipyblender Visualisierungen**

- Verschiedene Notebook-Plattformen wie JupyterLab, VS Code Notebook, Marimo und die minimalistische Alternative „Satyrn“ wurden erfolgreich mit Blender verbunden und getestet. https://x.com/kolibril13/status/1811007724970234173

- Ich habe ein Skript entwickelt, mit dem Netzwerke (Knoten und Kanten) direkt von Python aus in Blender geladen werden können (Abb. 4, Links). Diese Netzwerke werden mit dem Python Paket networkX berechnet, und anschließend in Blender verarbeitet. https://x.com/kolibril13/status/1809922563109257620

- Blender wurde zudem an die GPT-4 Vision API angebunden, um eine Zeichnung in eine 3D-Szene zu transformieren. https://x.com/kolibril13/status/1787396129883148387

- Ein weiteres Skript ermöglicht die interaktive Visualisierung von Datentabellen in Blender. Die Daten werden dabei mit dem auf Anywidget basierenden Paket "quak“ gefiltert und dynamisch aktualisiert. Abb. 4 (Rechts) zeigt beispielsweise einen geografischen Datensatz, welcher die durchschnittlichen Hauspreise in Kalifornien darstellt. Diese Informationen lassen sich in Echtzeit filtern, sodass beispielsweise nur Häuser in Küstennähe oder mit mehr als 100 Wohnungen angezeigt werden. https://github.com/kolibril13/ipyblender-experimental/blob/main/california_housing/california_housing.ipynb

- An dieser Stelle ein kurzer technischer Einblick: Die entwickelten Python-Skripte greifen auf die interne Datenstruktur eines Objekts in Blender zu, etwa auf Knotenpunkte, Kanten oder Flächen eines Würfels. Diese Datenstruktur kann im nächsten Schritt durch Koordinaten, Werte oder Text-Labels ergänzt werden. Anschließend werden diese Eigenschaften in Blender durch Geometrieoperationen, sogenannten „Geometry Nodes“, weiterverarbeitet. Zu den verfügbaren Operationen gehören Drehen, Skalieren, Transformieren, Verbinden, Kanten abrunden oder auch Duplizieren. Sobald sich die Datenstruktur eines Objekts ändert, werden die nachfolgenden Geometrieoperationen automatisch ausgeführt. Alle Skripte sind auf GitHub verfügbar. https://github.com/kolibril13/ipyblender-experimental

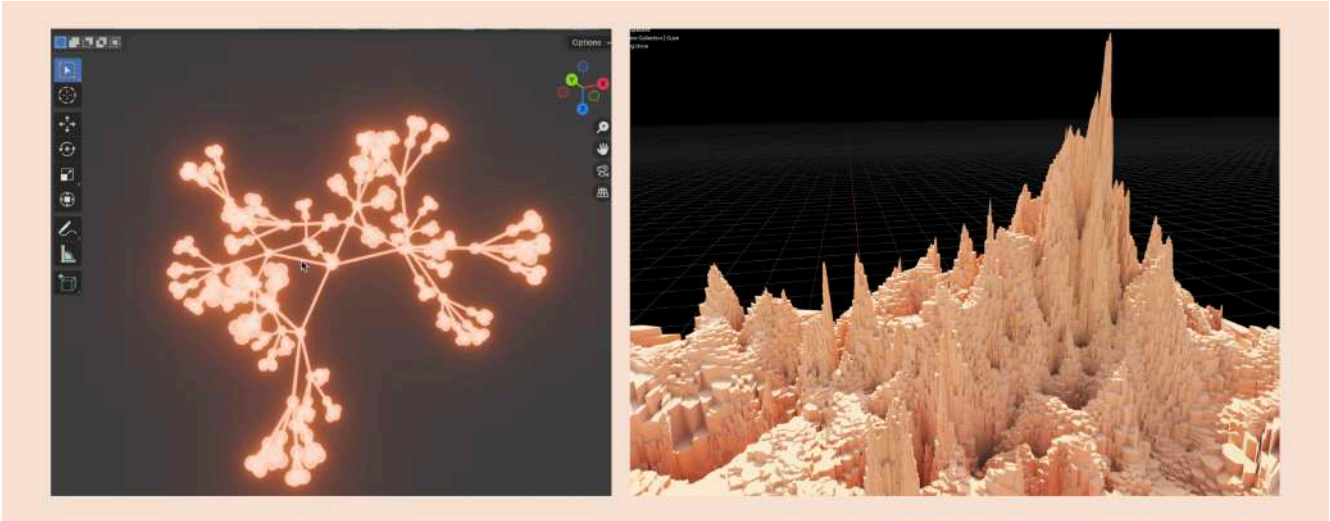


**Abb. 4:** Links - Visualisierung eines Netzwerkes, generiert mit networkX
Rechts - Darstellung eines geografischen Datensatzes mit dynamischer Aktualisierung

## Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Das Projekt richtet sich vor allem an EntwicklerInnen, WissenschaftlerInnen und JournalistInnen, die interaktive Visualisierungen in ihren Python-Notebooks einsetzen möchten. Der Hauptnutzen für die Zielgruppe liegt in der Automatisierung von Workflows und der Verbesserung iterativer Prozesse. Besonders für NutzerInnen, die an der Integration spezialisierter Frontend-Pakete in Notebooks interessiert sind, bietet das Projekt eine leicht zugängliche Plattform. In den Kommentaren auf Twitter wurde beispielsweise eine XYFlow-basierte Benutzeroberfläche zur Interaktion mit Graphen vorgeschlagen. Eine weitere Idee war, ein Whiteboard in Jupyter als Oberfläche zur Erfassung von Patientenzeichnungen bei kognitiven Untersuchungen zu nutzen. https://x.com/kolibril13/status/1793916530612453543

In Bezug auf die Interaktion mit Blender bin ich motiviert, meine Entwicklungen auf der Konferenz BlenderCon24 zu präsentieren und Workshops, etwa für Medienhäuser, anzubieten. Zukünftig könnte diese Methode zur Visualisierung verschiedenster Illustrationsprojekte eingesetzt werden – von der geografischen Platzierung neuer Solarzellen bis hin zur Animation des Temperaturanstiegs einer Region über mehrere Jahre. Außerdem möchte ich die erstellten Tools selbst verwenden, um Infografiken und Animationen zu erstellen, beispielsweise zur Visualisierung physikalischer Experimente.

## Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Ursprünglich hatte ich geplant, mein Tool auch über die Cloud zugänglich zu machen. Allerdings habe ich diese Idee verworfen, da der Aufbau einer entsprechenden Serverinfrastruktur erforderlich gewesen wäre, was mit erheblichem Aufwand verbunden gewesen wäre. Stattdessen habe ich die Entwicklung der Offline-Version des Tools priorisiert.

## Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Der Code zu dem Projekt ist auf Github zu finden: https://github.com/kolibril13/ipyblender-experimental

Außerdem poste ich regelmäßig Updates zu meinen Open-Source-Arbeiten auf X und Mastodon:
https://x.com/kolibril13
https://fosstodon.org/@kolibril13

Zusätzlich sind Web-Links zu den einzelnen Projektabschnitten im vorherigen Abschnitt beigefügt.

## Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Der Plan konnte eingehalten werden.

## Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Andere Institutionen hatten keinen großen Einfluss auf meine Arbeit und die Zielsetzung.

Besonderer Dank an folgende Menschen, die mich bei diesem Vorhaben unterstützt haben:
Trevor Manz - für die Entwicklung und den Austausch über anywidget
Maarten Breddels - für die Entwicklungsarbeit im Python Widget System
Myles Scolnick - für die Integration von Widgets in reaktive Python-Notebooks
Brady Johnston - für die Tipps bei der Integration von Python und Blender

Vielen Dank für die Unterstützung durch Superbloom, dem PrototypeFund-Team und die Förderung vom BMBF.