Bericht Abschlussprojekt

Softwaredesign

Wintersemester 2024

Ebene Mechanismen simulieren in Python

Bachelor - Mechatronik, Design und Innovation

Jahrgang: BA-MECH-23

Lehrveranstaltugsleiter: Julian Huber

Group: MECH-B-3-SWD-SWD-ILV

Verfassende: Schrott Maximilian & Naidr Jan

27. Februar 2025

Inhaltsverzeichnis

1 Aufgabenstellung

In dieser Projektarbeit (in Gruppen von maximal zwei Personen) soll ein Softwareprojekt realisiert werden. Die Versionsverwaltung erfolgt mittels `git` und das Repository wird auf GitHub gehostet. Ziel ist die Entwicklung einer Python-Anwendung, die objektorientiert programmiert und über eine Web-Benutzeroberfläche (Streamlit) bedient wird. Die Anwendung ermöglicht die Definition und Simulation der Kinematik beliebig definierbarer Mechanismen (unter vorgegebenen Einschränkungen).

1.1 MINIMALANFORDERUNGEN

Folgende Anforderungen müssen für eine positive Bewertung berücksichtigt werden:

- **Softwareprojekt:**
  - Umsetzung in Python unter Anwendung objektorientierter Programmierung.
  - Versionskontrolle mit `git` und Hosting auf GitHub.
- **Web-Anwendung:**
  - Entwicklung einer Anwendung mit Web-UI (Streamlit).
  - Simulation von Mechanismen und deren Kinematik (Details siehe unten).
- **Software-Dokumentation:**
  - Bereitstellung einer `requirements.txt`-Datei mit allen benötigten Packages.
  - Eine `README.md`-Datei im Repository mit Installations- und Ausführungsanleitung.
- **Projekt-Dokumentation:** Gewählt:
  - Kurzer Bericht (PDF) mit Angaben zu umgesetzten Erweiterungen, UML-Diagrammen der Softwarestruktur, Quellenangaben etc.
- **Mechanismen und Kinematik:**
  - Definition beliebiger Mechanismen (unter Berücksichtigung der vorgegebenen Einschränkungen).
  - Berechnung der Positions-Kinematik für Winkel im Bereich von 0° bis 360°.
  - Visualisierung des Mechanismus, seiner Kinematik und der Bahnkurven der Punkte.
  - Möglichkeit zum Speichern der Bahnkurven (z. B. als $x(\theta)$ – $y(\theta)$ -Koordinaten im CSV-Format oder ähnlich).
  - Speicherung und Laden des Mechanismus samt zugehöriger Kinematik.
- **Optimierungsansatz:**
  - Formulierung der Kinematik als Optimierungsproblem, bei dem die Längenfehler der Mechanismenglieder minimiert werden.
- **Validierung und Test:**
  - Überprüfung der Validität des Mechanismus.
  - Test der Implementierung am Beispiel des „Strandbeest“ bzw. eines seiner Beine – Ergebnisse in der Dokumentation darlegen.
- **Deployment:**
  - Deployment der Anwendung mit Streamlit.

2 Hauptteil

2.1 WIE FUNKTIONIERT DIE SIMULATION?

Die gesamte App, und damit auch die Simulation selbst, läuft über den Link `https://strandbeest-simulation.streamlit.app/`. Der gesamte Prozess läuft über die Streamlit Community Cloud, die es ermöglicht, dass die Anwendung 24 Stunden am Tag, 7 Tage die Woche online verfügbar ist.

Die Streamlit-App bietet eine komfortable Möglichkeit, diese Simulation zu steuern. Mit editierbaren Tabellen und Feldern können die Parameter des Mechanismus direkt geändert, gelöscht oder hinzugefügt werden. Alles wird dann direkt in einer Datenbank gespeichert und die aktualisierten Daten werden für die neue Berechnung verwendet. Vor Beginn der Berechnung wird es auch geprüft, ob der Mechanismus als gültig eingestellt ist. Ist dies nicht der Fall, wird eine Fehlermeldung angezeigt und die Parameter müssen angepasst werden.

Die Koordinaten der vorhandenen Punkte können nach Bedarf angepasst werden. Dabei können die Punkte unterschiedliche Eigenschaften haben → Fixiert, Driver, Pivot, Schubführung_X und Schubführung_Y. Alle beliebigen Kombinationen sind natürlich nicht erlaubt, denn dann könnte der Mechanismus keine sinnvolle Bewegung ausführen. Die Funktion Valid-Check prüft, ob nur freigegebene Kombinationen und Einstellungen aktiv sind.

Zu beachten ist dabei, dass die Kreisbewegung mit einen Winkel von 0° starten muss, da ansonsten die Berechnung fehlerhaft wäre.

Durch Auswahl von bestehenden Punkten lassen sich auch neue Stangen (Verbindungen zwischen Punkten) erstellen. Die Punkte werden in einer Liste angezeigt, und der Nutzer kann den Startpunkt, den Endpunkt und den Namen der Stange auswählen. Das Ganze wird dann wieder in der Datenbank gespeichert und auf Gültigkeit geprüft. Wenn alles in Ordnung ist, werden die neuen Verbindungen zur Berechnung des ebenen Mechanismus verwendet.

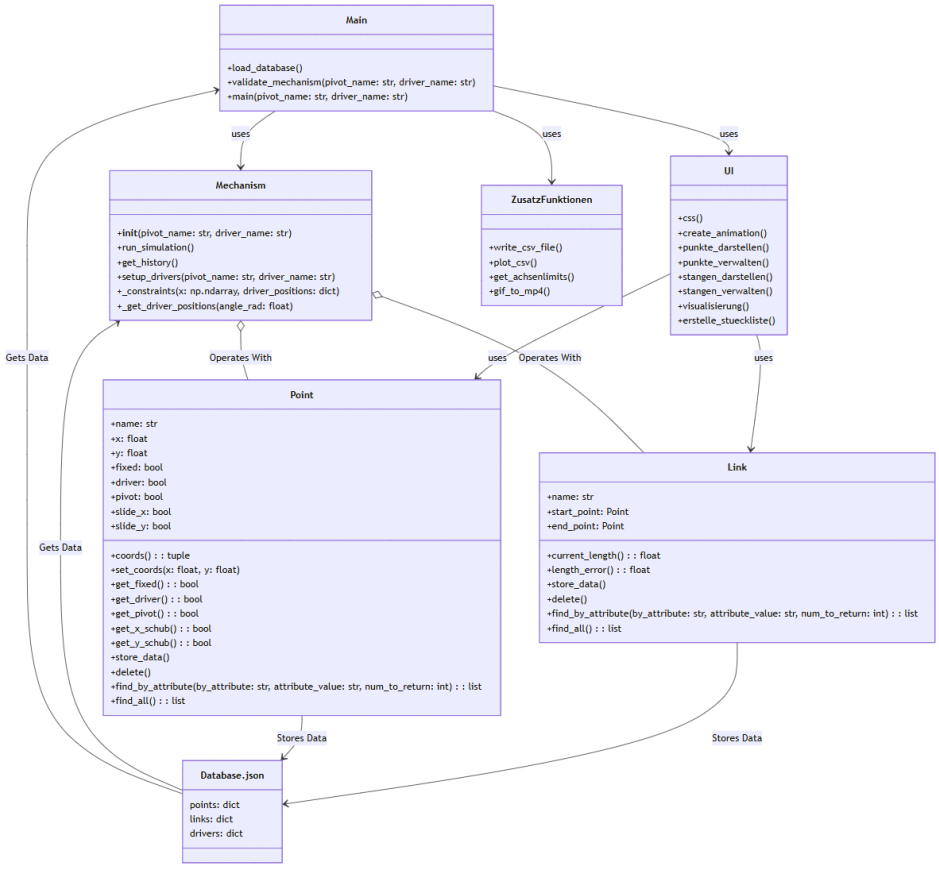

Abbildung 2.1: App-Struktur visualisiert mit UML-Diagramm

Das obige Diagramm Figure 2.1 zeigt, wie die App und der Code strukturiert sind. Die Hauptklasse Mechanism, in der die Berechnung stattfindet, basiert auf zwei Unterklassen Point und Link. Database.json dient als Speicherort für die Parameter des Mechanismus. Die Streamlit- Oberfläche wird in der Datei *ui.py* erstellt. Alles wird dann in der Datei *main.py* aufgerufen, um das gewünschte Ergebnis zu erhalten.

2.2 ERWEITERUNGEN

Neben den Minimalanforderungen wurden folgende Erweiterungen implementiert:

- **Visualisierung der Längenfehler:** Darstellung der Abweichungen der tatsächlichen Gliederlängen von den Zielmaßen als Funktionen des Winkels.
- **Animationserstellung:** Möglichkeit, die Simulation als Video oder GIF zu animieren, zu speichern und herunterzuladen.
- **Erweiterte Visualisierungs-Overlays:** Einblendung zusätzlicher Informationen (z. B. Bahnkurve, Namen) direkt in die Visualisierung.
- **Erweiterte Kinematiklösungen:** Unterstützung von Kinematiken, bei denen ein Punkt einen zusätzlichen Freiheitsgrad besitzt (z. B. Schubkurbeln).
- **Automatisierte Stückliste:** Generierung einer Stückliste für ausgewählte Komponenten wie Gestänge, Antriebe und Gelenke.

Erklärung der Erweiterungen:

1. Die Längenfehler der Mechanismusglieder werden als Funktion des Drehwinkels θ geplottet, um Abweichungen von den Sollwerten zu analysieren. Die Visualisierung zeigt, wie sich die Fehler über den Bewegungszyklus entwickeln, und hilft, Ungenauigkeiten im Design zu erkennen und zu optimieren. Das Diagramm zeigt, wie stark und auf welche Weise jede Stange betroffen ist.
2. Die Simulation wird als Video und als GIF animiert. Zudem kann man die Anwendung herunterladen, sodass Bewegungsabläufe und dynamische Effekte anschaulich dokumentiert und geteilt werden können. Außerdem wird Animation erst verändert wenn man die Berechnung neu durchführen lässt.
3. Überlagernde Beschriftungen, wie z. B. Namen oder Bahnkurven, werden in Echtzeit auf die Simulation gelegt, um detaillierte Einblicke in den Mechanismus zu bieten. Die Beschriftungen erscheinen wenn man den ÖverlayButton drückt und die Berechnung neu ausführt.
4. Punkte, die die Eigenschaft slide_x erfüllen, können sich nur in x-Richtung bewegen, ihre y-Koordinate bleibt konstant. In analoger Weise ist die x-Koordinate für slide_y fest. Diese Bedingungen werden während der Optimierung überprüft, um sicherzustellen, dass sich die Punkte nur innerhalb der zulässigen Freiheitsgrade bewegen. Diese Punkte können dann als Schubkurbel gezeichnet werden.
5. Für definierte Mechanismen wird eine Stückliste erstellt, die alle relevanten Komponenten (Gestänge, Antriebe, Gelenke) auflistet und so die Planung und Montage unterstützt.

2.3 VALIDIERUNG DES MECHANISMUS

Die Validierung der Implementierung wurde durch die Simulation eines Mechanismus basierend auf dem *Strandbeest* von Theo Jansen überprüft, insbesondere anhand eines seiner Beine. Dabei wurde folgendermaßen vorgegangen:

1. Erstellung einer vollständigen Simulation eines *Strandbeest*-Beins, die Koordinaten wurden selbst interpretiert da online keine gefunden wurden.

2. Überprüfung der berechneten Bahnkurve der Fußpunkte mit bekannten Referenzwerten.
3. Darstellung der Bewegung mittels animierter Visualisierung zur anschaulichen Überprüfung.

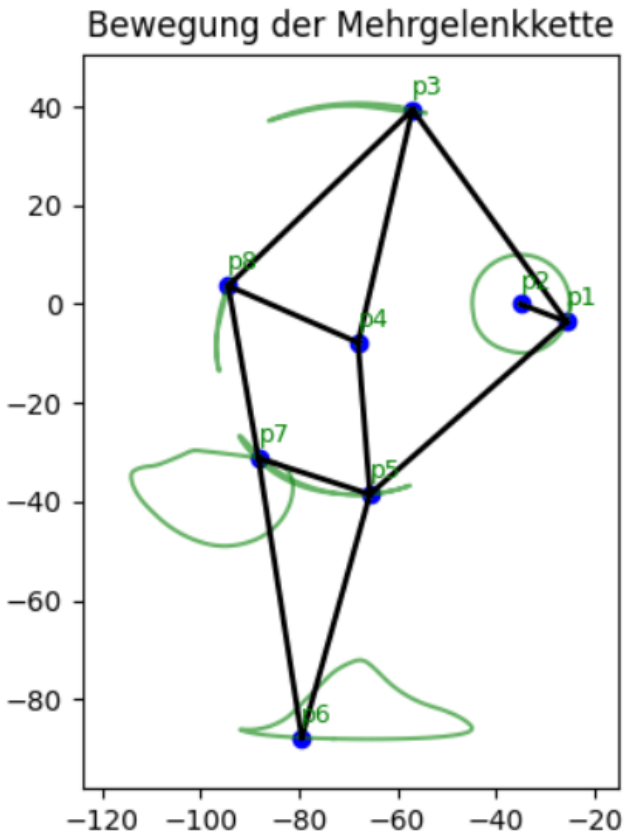

Abbildung 2.2: Visualisierung der Mechanismus-Validierung anhand der Simulation eines *Strandbeest*-Beins.

Im Vergleich der Bahnkurve von "Punkt6"(p6), in Abbildung 2.2 dargestellt, mit den bekannten Referenzwerten, zeigt die simulierte einen anderen Verlauf. Dies ist aber darauf zurückzuführen, dass nicht die gleichen Stablängen verwendet wurden. Ansonsten war die Validierung erfolgreich und bestätigt die korrekte Funktionsweise der Implementierung. Das Ergebnis zeigt, dass die entwickelte Anwendung zuverlässig Mechanismen modellieren und simulieren kann.

2.4 VERWENDUNG VON KI-TOOLS

KI-Tools (wie z.B. ChatGPT) wurden verwendet, um:

- Ordnung in den Code zu bringen, also zur Verbesserung und Strukturierung des Codes, sodass dieser leichter verständlich ist.
- Erklärungen zu einigen Fehlermeldungen zu liefern.
- Copilot von Visual Studio Code wurde verwendet, um das Programmieren von ähnlichen Stellen viel effizienter zu machen, da man sich viele Tastenschläge spart.

Abbildungsverzeichnis