IU Internationale Hochschule
Weiterbildung: “Software Engineering - Python”
Modul : DLMCSPSE01_D - Projekt: Software Engineering
Tutor: Prof. Dr. David Kuhlen

Prüfungsleistung: Portfolio

Architekturdokument

Eingereicht am 16.09.2025.

Verfasser:
Djahan Bayrami Latran
Denninger Straße 198
81927 München
E-Mail: djahan.latran@gmail.com
Matrikelnummer: UPS10672478

1. Technologieübersicht:

Verwendete Programmiersprache:

- **Python**:
  Für die Umsetzung des Projekts wurde die Programmiersprache Python verwendet. Python eignet sich besonders für die schnelle Entwicklung von Prototypen und zeichnet sich durch eine umfassende Standardbibliothek sowie viele weitere Drittanbieter Bibliotheken aus. Durch die große und engagierte Community stehen eine Vielzahl an Ressourcen und Dokumentationen online zur Verfügung.

Verwendete Bibliotheken:

- **Pygame**:
  Die Bibliothek “Pygame” wird als Basis für die grafische Darstellung der Applikation verwendet. Sie ist dafür ausgelegt, eine effiziente Darstellung von zweidimensionalen Bildern und Animationen zu ermöglichen. Dank dieser Eigenschaften eignet sich “Pygame” besonders zur Visualisierung der Algorithmen im Rahmen der Applikation.
- **Pygame_gui**:
  “Pygame_gui” erweitert “Pygame” um vorgefertigte GUI-Elemente zur User-Interaktion, wie z.B. Schaltflächen und Schieberegler. Die Integration von GUI-Elementen aus “Pygame_gui” in die “Pygame”-Oberfläche ist deshalb simpel und effektiv. Auf die Verwendung von gängigen UI-Bibliotheken wie “Tkinter”, “PyQt” und “PySide” wurde aus diesem Grund verzichtet.
- **Pygments**:
  Pygments wird für die Darstellung von Texten in einem Quellcode-Stil innerhalb der Benutzeroberfläche genutzt. Durch Funktionen der Bibliothek werden farbliche Hervorhebungen im Text ermöglicht. Das optische Erscheinungsbild ähnelt Quelltext aus einer Entwicklungsumgebung.
- **PyYAML**:
  PyYAML wird zur Verarbeitung von YAML-Dateien und deren textuellen Inhalt genutzt. Texte werden aus den YAML-Dateien eingelesen und anschließend in der Benutzeroberfläche angezeigt. PyYAML ermöglicht somit die Verwaltung von textuellen Inhalten außerhalb des Quellcodes.
- **Python Standard Library** (collections, abc, time, os, random):
  In der Anwendung wurden unterschiedliche Module der Python Standard Library verwendet. Mit Hilfe von “time” werden die Aktualisierungszyklen der Anwendung gesteuert. Mit “Abc” wurde eine abstrakte Basisklasse implementiert, die als Grundlage für weitere Unterklassen dient. “Random” wurde genutzt, um Zufallswerte

zu generieren, die als Eingabewerte der Algorithmen fungieren. Zusätzlich wurde “collections” für die Integration von effiziente Datenstrukturen und “os” zur Ausführung von datei- und verzeichnisbezogene Operationen verwendet.

2. Architekturübersicht:

Architekturmodell:
Es wurde eine modularisierte Schichtenarchitektur im Sinne des MVC-Designs (Model-View-Controller) umgesetzt. Durch das Separieren der Komponenten in unabhängige Module wird eine einfachere Erweiterung, Wartung und Wiederverwendbarkeit gewährleistet. Neue Algorithmen oder GUI-Komponenten können leichter und effizienter integriert werden. Es entsteht eine saubere Trennung von Daten, Anzeige und Logik.

Komponenten:

- Model:
    - Enthält die Klassen “AppStates”, “Board”, “Parameter” und die Algorithmen Klassen (z.B. “BubbleSort” oder “BinarySearch”).
    - Trägt Verantwortung über den Status der Anwendung. (z.B. der momentane Zustand der zu sortierenden Eingabewerte eines Sortieralgorithmus.)
- View:
    - Fügt sich zusammen aus den Klassen “GuiManager”, den unterschiedlichen “Panel”-Klassen, sowie den in den “Panel”-Klassen verwendeten GUI-Elementen (z.B. “Button”, “TextWindow”, “Slider”).
    - Ist verantwortlich für die visuelle Übertragung der Applikation und leitet Eingabe-Interaktionen durch den Nutzer weiter an den Controller.
- Controller:
    - Besteht aus der Klasse “AppController”.
    - Vermittelt zwischen Model und View und ist für die Logik der Anwendung verantwortlich. Besitzt Komponenten wie “AppStates” und “FileReader”, verwaltet Zustände und reagiert auf Benutzeraktionen, die vom View übermittelt werden.

3. Struktur:

UML-Klassendiagramm:

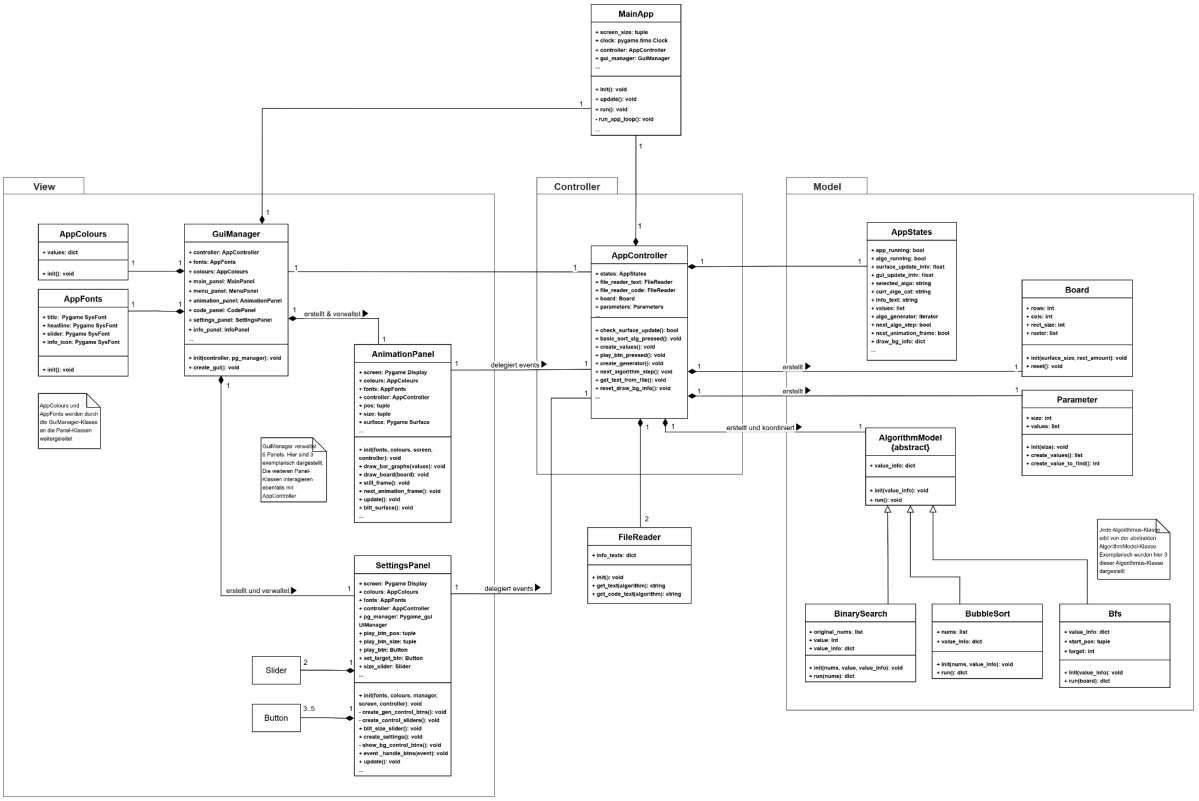

Abbildung 8: Klassendiagramm der Anwendung, erstellt mit draw.io

MainApp:

- Führt die Hauptschleife der Anwendung aus und initialisiert die beiden zentralen Komponenten “GuiManager” und “AppController”.

AppController:

- Ist zuständig für die Logik und Steuerung der Applikation. Erstellt die Algorithmen-Klassen und koordiniert den Ablauf und die Visualisierung. Besitzt eine “AppStates” Instanz und ändert deren Attribute, um so den Zustand der Anwendung zu steuern.

GuiManager:

- Initialisiert und hält die verschiedenen Panel-Klassen.

AlgorithmModel:

- Ist eine abstrakte Klasse, von der alle Algorithmus-Klassen erben. Die vererbte run()-Methode muss von den unterschiedlichen Algorithmen-Klassen als Generator implementiert werden. Dadurch können die Daten schrittweise visualisiert werden.

AppStates:

- Speichert die unterschiedlichen Zustände. Enthält keine Methoden, sondern kennzeichnet sich durch eine erhöhte Anzahl an Attributen. Dadurch werden z.B. der momentan gewählte Algorithmus, die jeweilige Kategorie und der Ausführungsstatus (pausiert, läuft, zurücksetzen, etc.) gespeichert.

Panels (z.B. AnimationPanel, MenuPanel, SettingsPanel, usw.):

- Die grafische Benutzeroberfläche ist in verschiedene Bereiche unterteilt, die durch die Panel-Klassen repräsentiert werden. Sie enthalten unterschiedliche GUI-Elemente. AnimationPanel ist zuständig für die visuelle Darstellung der Algorithmen, SettingsPanel für die Konfiguration von Parametern, MenuPanel für die Auswahl des Algorithmus, InfoPanel für die Bereitstellung von zusätzlichen Informationen in Textform und CodePanel für die Darstellung des Codes.

FileReader:

- Lädt YAML-Dateien mit textuellen Inhalten, die anschließend im CodePanel und InfoPanel dargestellt werden.

Parameter und Board:

- Hilfsklassen, die für die Erstellung der Eingabeparameter, sowie für die Gitterstruktur der Graphenalgorithmen dienen.

Button, Panel, Slider, TextWindow, Scrollcontainer:

- Wrapper zu vorgefertigten pygame_gui-Elementen.

3. Verhalten:

Ablauf Algorithmus Auswahl:

Im folgenden Sequenzdiagramm wird der Prozess der Auswahl eines Algorithmus in der Anwendung und die darauf folgende interne Verarbeitung dargestellt. Hier wird ersichtlich, wie der Controller (mittig dargestellt) zwischen dem View und Model koordiniert.

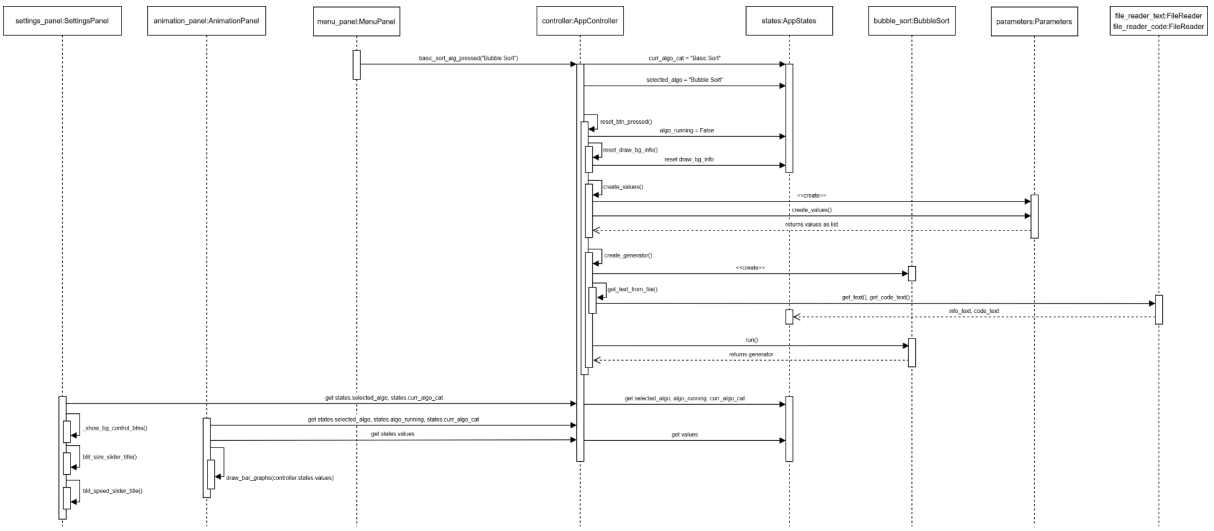

Abbildung 9: Sequenzdiagramm zum Ablauf “Algorithmus Auswahl”, erstellt mit draw.io

Ablauf Algorithmus starten:

In diesem Sequenzdiagramm wird der interne Ablauf beim Starten eines Algorithmus in der Anwendung veranschaulicht. Auch hier ist der Controller zentral als die verantwortliche Komponente für die Logik der Applikation dargestellt.

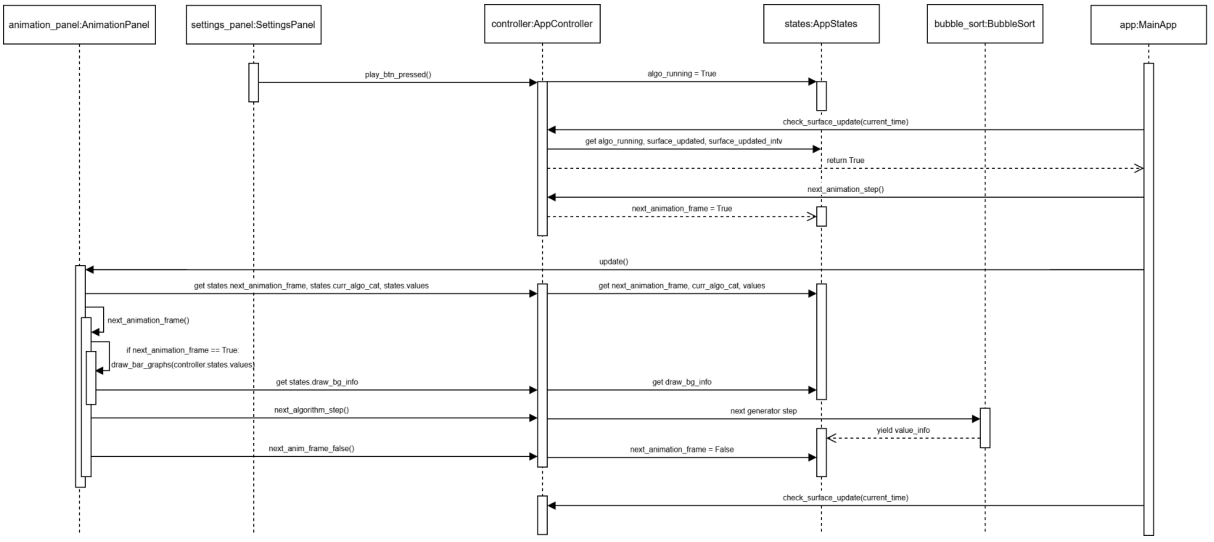

Abbildung 10: Sequenzdiagramm zum Ablauf “Algorithmus starten”, erstellt mit draw.io