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

Prüfungsleistung: Portfolio

Architekturdokument

Eingereicht am 11.06.2025.

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

1. Technologieübersicht:

Verwendete Programmiersprache:

- **Python**:
  Hier fiel die Wahl auf Python, da ich durch meine Python spezifische Weiterbildung in dieser Sprache über tiefer gehende Kenntnisse verfüge, als bei anderen objektorientierten Sprachen. Außerdem eignet sich Python hervorragend zur schnellen Implementierung von Prototypen und ist durch seine große Community und die vielfältige Auswahl an Bibliotheken für unterschiedlichste Anwendungszwecke geeignet.

Verwendete Bibliotheken:

- **Pygame**:
  Wird als Basis für die grafische Darstellung der Applikation verwendet. Grafische 2D Animationen, sowie Benutzerinteraktionen lassen sich mit Pygame einfach und effizient umsetzen. Für die visuelle Darstellung der Algorithmenfunktionsweisen ist die Bibliothek deshalb bestens geeignet.
- **Pygame_gui**:
  Diese Bibliothek erweitert Pygame um vorgefertigte GUI-Elemente zur Interaktion, wie bspw. Buttons und Sliders. Die Integration in das Pygame Window ist deshalb sehr einfach, weshalb auf die Verwendung von anderen gängigeren UI-Bibliotheken wie bspw. “Tkinter”, “PyQt” oder “PySide” verzichtet wurde.
- **Pygments**:
  Pygments fand Verwendung zur Formatierung von Texten, die auf der Benutzeroberfläche als “Code” angezeigt werden. Mit Funktionen der Bibliothek wird der Text so eingefärbt, dass er optisch so wirkt wie Code aus einer Entwicklungsumgebung.
- **PyYAML**:
  Wird verwendet um Texte aus Yaml-Dateien zu lesen, die anschließend in der Applikation angezeigt werden.
- **Python Standard Library** (collections, abc, time, os, random):
  Hier wurden unterschiedlichste Module verwendet wie beispielsweise “time”, um die vergangene Zeit seit Start der Anwendung zu erhalten, was wichtig für die Aktualisierungsintervalle ist. “ABC” kam zum Einsatz bei der Implementierung einer abstrakten Klasse und “random” bei der Kreation von randomisierten Eingabewerten.

2. Architekturübersicht:

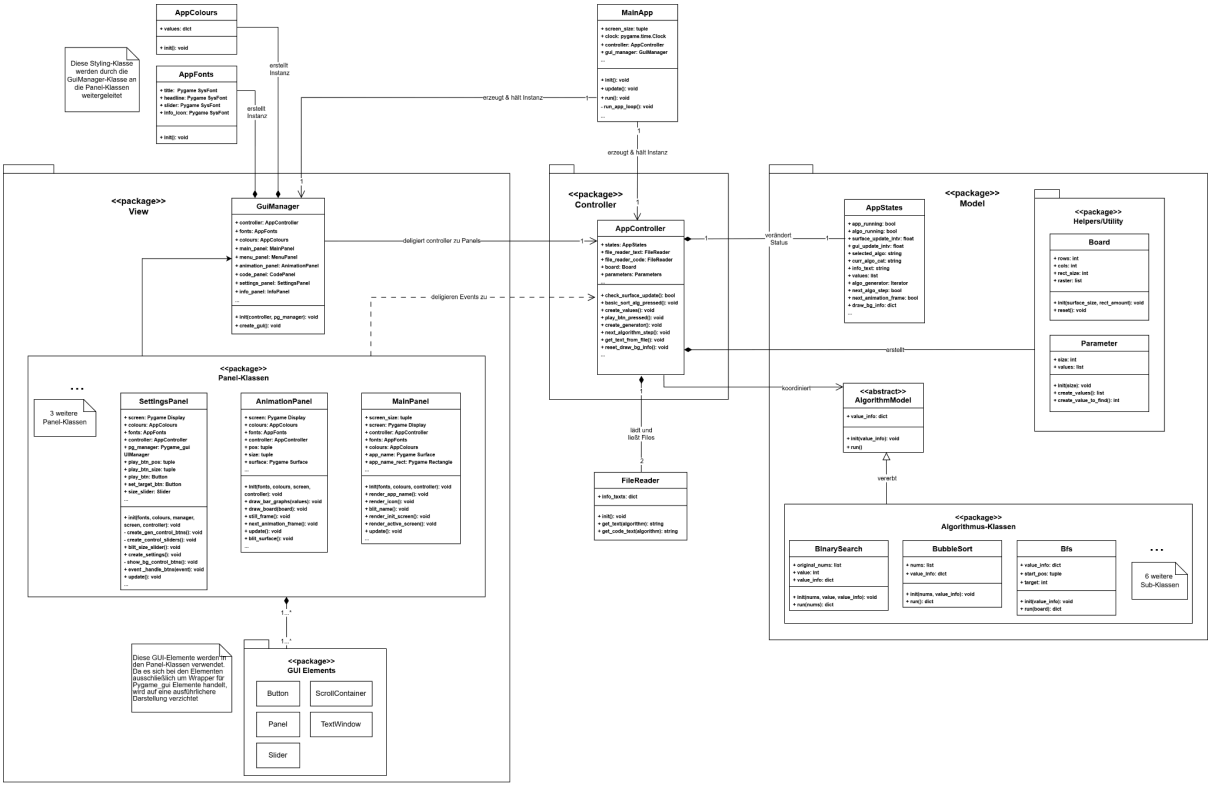
Architekturmodell:
Es wurde eine modularisierte Schichtenarchitektur im Sinne des MVC-Designs (Model-View-Controller) umgesetzt. Durch das Separieren der Komponenten in unabhängige Module wird eine einfachere Erweiterung, Wartung und Wiederverwendbarkeit gewährleistet. Neue Algorithmen oder GUI-Komponenten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

Komponenten:

- Model:
    - Enthält die Klassen “AppStates”, “Board”, “Parameter” und die Algorithmenklassen wie bspw. “BubbleSort”, “BinarySearch”, usw. .
    - 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 “Pane”l-Klassen verwendeten GUI-Klassen, wie “Button”, “TextWindow”, “Slider”, usw. .
    - 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 somit Zustände und reagiert auf Benutzeraktionen, die er vom View erhält.

3. Struktur:

UML-Klassendiagramm:



MainApp:

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

AppController:

- Diese Klasse ist zuständig für die Logik und Steuerung der Applikation. Sie erstellt die Algorithmen-Klassen und koordiniert den Ablauf und die Visualisierung derer. Sie besitzt eine “AppStates” Instanz und ändert deren Attribute, um so den Zustand der Anwendung zu steuern.

GuiManager:

- GuiManager initialisiert und hält die verschiedenen Panel-Klassen.

AlgorithmModel:

- Dies ist eine abstrakte Klasse, von der alle Algorithm-Klassen erben. Die vererbte run()-Methode muss von den unterschiedlichen Algorithmen-Klassen als Generator implementiert werden. So können im Endeffekt die Daten schrittweise visualisiert werden.

AppStates:

- Diese Klasse speichert unterschiedlichste Zustände. Sie enthält keine Methoden, sondern kennzeichnet sich durch eine erhöhte Anzahl an Attributen. Dadurch wird beispielsweise 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. Diese Klassen 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 Code's.

FileReader:

- Lädt und ließt Texte aus YAML-Dateien, 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.

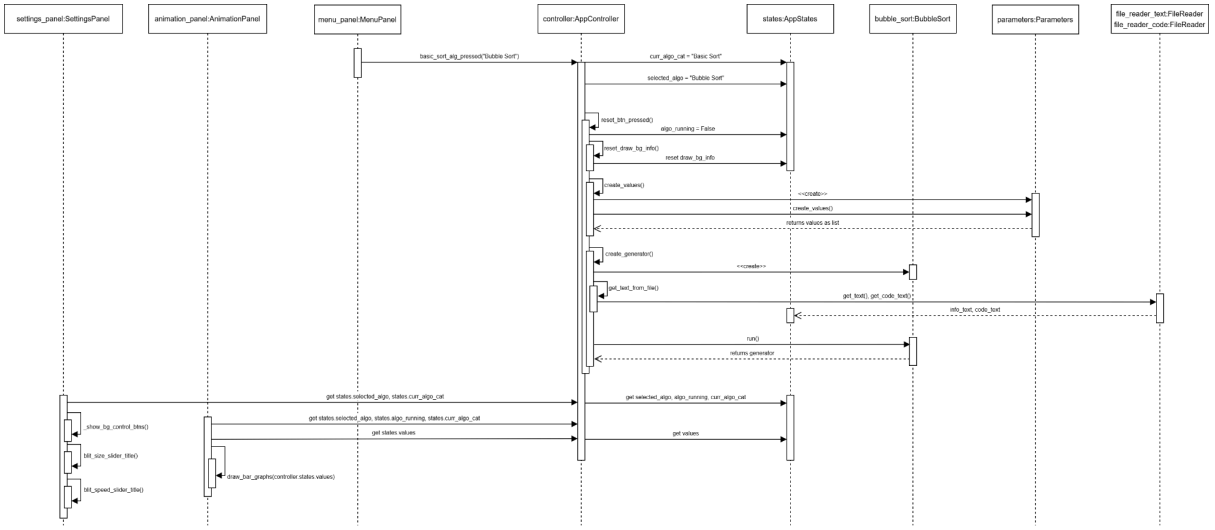
Button, Panel, Slider, TextWindow, Scrollcontainer:

- Klassen die als Wrapper für vorgefertigte pygame_gui-Elemente dienen.

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.



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.