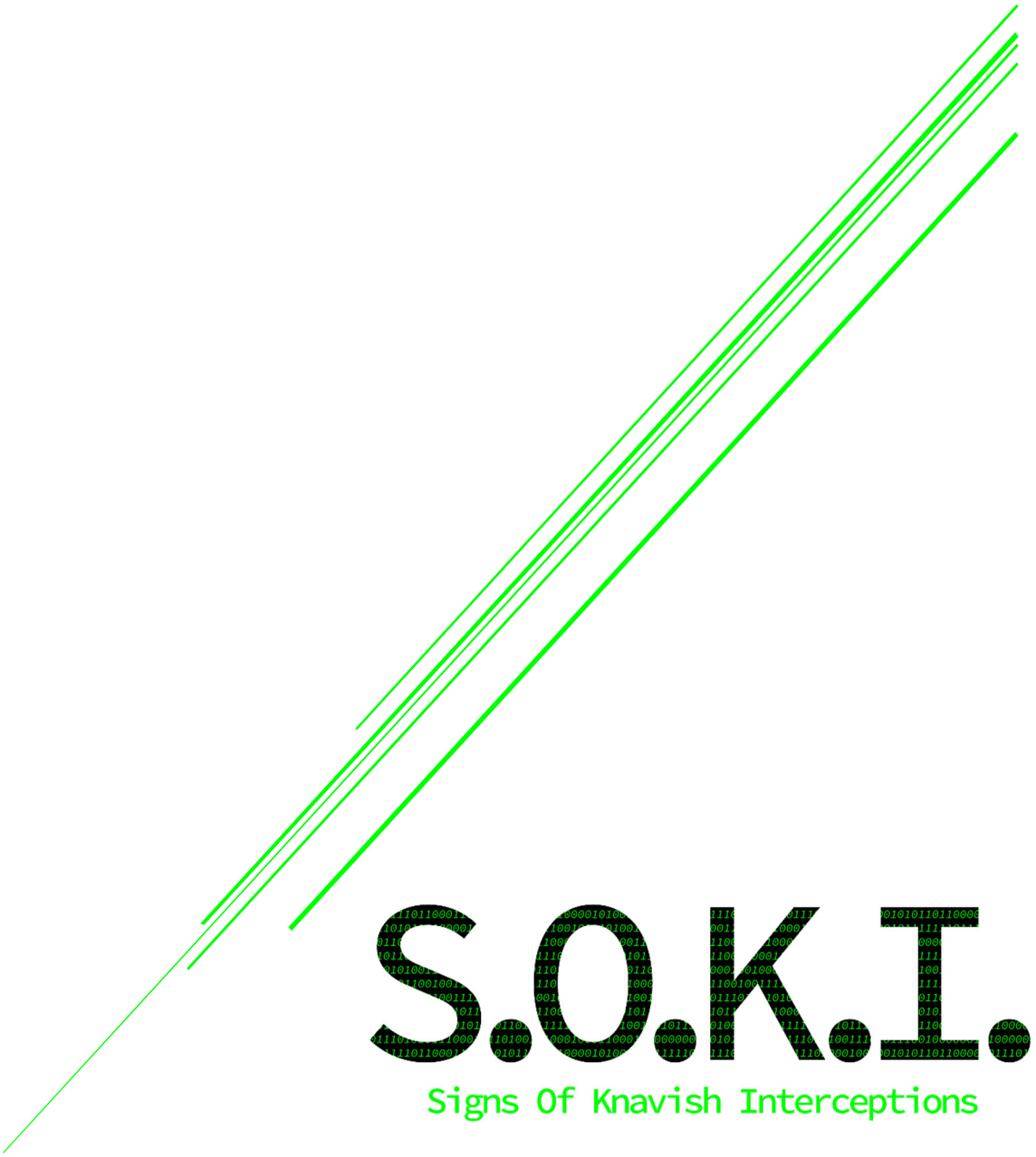
ENTWICKLERDOKUMENTATION

Erklärungen und Möglichkeiten im Verlauf von SOKI



Inhaltsverzeichnis

Einleitung

In diesem Dokument sei erläutert, wie der Leser die Laufzeitumgebung für die erfolgreiche Inbetriebnahme des Programmes einrichten kann. Ferner sind Informationen zu der benötigten Entwicklungsumgebung gelistet, mit welchen der Leser selbst das Programm fortsetzen oder aber bearbeiten kann. Sollte der Leser indes rein an der Architektur des Programmes interessiert sein, so findet er im Kapitel “Umsetzung der Anwendung” hilfreiches Wissen zu eben diesem Schwerpunkt. Im letzten Teil dieser Dokumentation finden sich Ideen für die Weiterentwicklung des Programmes und die Planung des Projektes.

Laufzeit und Entwicklungsumgebung

Einrichten der Laufzeitumgebung

- *Java OpenJDK 17*
  - *Maven, Version 3.8.1*
- *JavaFX 17.0.1*
- *Windows 10/11*

Es wird OpenJDK 17 mit JavaFX und Maven benötigt, um das Programm zu starten

Einrichten der Entwicklungsumgebung

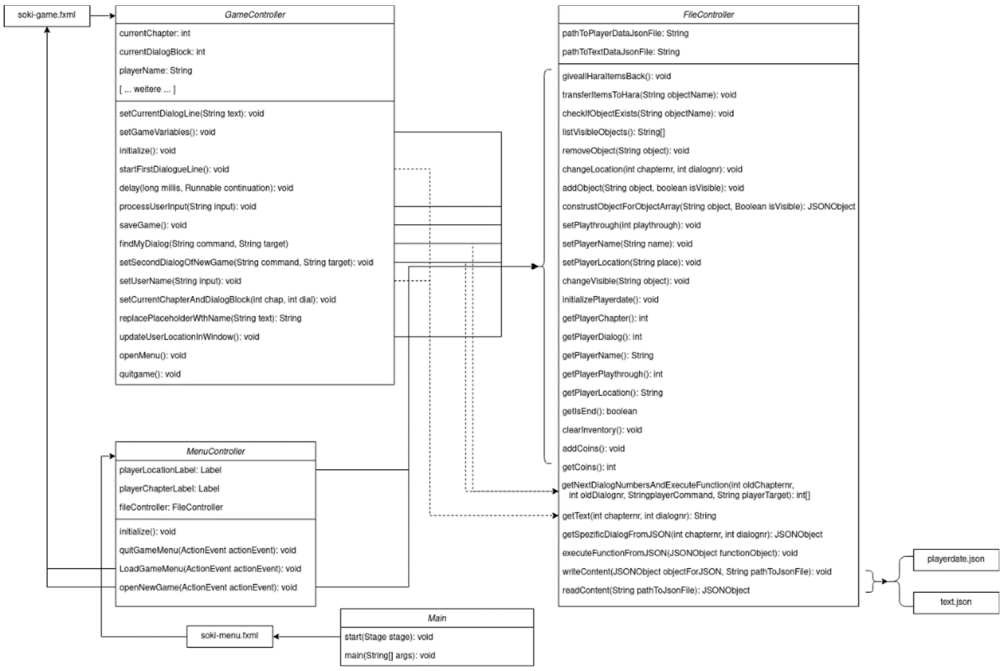
Es wird eine entsprechende IDE vorausgesetzt, welche den Anforderungen (siehe oben) Genüge leistet. Als Beispiel sei hier IntelliJ IDEA zu nennen, es sind aber selbstverständlich auch andere Programme wie z.B. Eclipse nicht auszuschließen. IntelliJ lässt sich auf diversen Betriebssystemen installieren, es seien hier z.B. Windows 10 oder auch Linux genannt.

Für Gestaltung des User Interfaces lassen sich Programme wie SceneBuilder zur Hilfe nehmen.

Im Idealfall sei es mit dem Programm möglich das Projekt S.O.K.I von dem offiziellen GitHub Repository zu klonen:

-> https://github.com/Spottedstorm23/team-soki

Umsetzung der Anwendung



Das Programm ist im Wesentlichen in 3 Controller aufgeteilt, welche das Programm steuern.
Die Main-Klasse führt dabei direkt zum Start des Programmes die soki-menu.fxml auf, dessen Eingaben über den MenuController verarbeitet werden.
Der MenuController verwaltet das erste Interface, welches ebenda als “Menü” zu bezeichnen ist. Dieser Controller bezieht bereits zum Anzeigen diverser Informationen Daten aus dem FileController (zu welchem später genauere Erläuterungen folgen).
Sollte der Benutzer ein neues Spiel starten oder ein vorhandenes laden, öffnet er damit unweigerlich das Spiel selbst, welches über soki-game.fxml repräsentiert und durch den GameController umgesetzt ist. Während des Spiels ist der GameController inn konstanter Kommunikation mit dem FileController, welcher Informationen zum Spieler und zu den Dialogzeilen ausgibt, ferner aber auch neue Spielstände und Attribute zu dem Spieler abspeichern kann. Hierfür greift der FileController auf die beiden JSON-Dateien playerdate.json und text.json zu.

Es sei an dieser Stelle angemerkt, dass sich ein hochauflösendes Bild des UML-Diagrammes, wie es sich oben vermerken lässt, im Appendix befindet.

Indes sind nähere Informationen insbesondere zum hierarchischen und strukturellen Aufbau der text.json direkt im Quellcode in der Datei “erklärungTextBlöcke.txt” zu finden. Ferner befindet sich ein handschriftlicher Entwurf der Struktur ebenfalls im Appendix.

Die funktionalen Teile des Programmes sind in der Hochsprache Java geschrieben. Gestaltung und UI sind mithilfe von Java FX umgesetzt. Die Speicherung erfolgt mittels der JSON.

Erweiterungsmöglichkeiten & nicht umgesetzte Funktionen

Es sei hier eine Liste der Funktionen aufgeführt, welche entweder in dieser Form oder gänzlich nicht implementiert sind.

- Erfolge, welche im Verlauf der Handlung für bestimmte Errungenschaften vergeben werden. Im selben Zusammenhang ist auch eine Übersicht für die visuelle Betrachtung dieser denkbar.
- Spracheinstellungen, welche über das Hauptmenü auswählbar sind
- Sogenannte “Easter-Eggs”, welche Anspielungen beinhalten, welche nur durch Wissende interpretierbar sind oder aber unerwartete Interaktionen darstellen.
- Eine Mehrspielerfunktion, welche da bedeutet, dass mehrere Spieler hintereinander (nicht zu verwechseln mit paralleler, kooperativer oder gar feindlicher Interaktion) das Spiel benutzen können. Möchte man ein Spiel laden, muss der entsprechende Benutzer ausgewählt werden. Entwürfe für die Funktion lassen sich in FileController_old finden, welcher aber in der aktuellen Version nicht verwendet wird.

Projektplanung

Zu Beginn des Projektes wurden die Aufgaben des Projektes fair aufgeteilt. Nach der Erstellung eines Git-Repositories begannen die Entwickler*innen eigenständig ihre Aufgaben abzuarbeiten. Dabei entstand diese Zeittabelle:

Datum	Julie	Selina	Cora	Lukas
Gesamtstunden:	52,75	44,17	46,42	49,33
29.03.2022	60	60	60	60
30.03.2022	90	90	90	90
31.03.2022			60	
01.04.2022	150	150	150	150
02.04.2022				240
03.04.2022		150		
06.04.2022	60	60	90	60
08.04.2022	240	290	250	260
09.04.2022		80		
13.04.2022	240	210	210	225
14.04.2022	140	120	120	210
24.04.2022	60		60	
29.04.2022				90
02.05.2022		60	270	60
03.05.2022	240	240	240	255
04.05.2022				90
05.05.2022				105
06.05.2022	15		15	15
08.05.2022				240
13.05.2022	60	60		
17.05.2022	30			
18.05.2022	60			
21.05.2022		180	180	
22.05.2022			120	
23.05.2022		120	150	
24.05.2022			240	
28.05.2022			180	
29.05.2022	120	60	60	
30.05.2022	30			
01.06.2022	90			
02.06.2022	240	180	180	150
04.06.2022	300			
05.06.2022	240	120		240
06.06.2022	700	420	60	420

Es sei angemerkt, dass diese Tabelle möglicherweise unvollständig ist, da einige Personen vergaßen, ihre Stunden einzutragen.

In der folgenden Tabelle werden die Aufgaben und die Beteiligung der Entwickler*innen dargestellt.

Aufgabe	Beteiligte Entwickler*innen
Entwurf des Logos	Cora V. Stokowsky
User Interface und Design	Selina Zesewitz-Thiel
Verfassen des Pflichtenhefts	Selina Zesewitz-Thiel, Cora V. Stokowsky, Lukas Keinert, Julie L. Haupt
Verfassen der Entwicklerdokumentation	Lukas Keinert, Julie L. Haupt
Verfassen des Nutzerhandbuchs	Selina Zesewitz-Thiel, Cora V. Stokowsky
Entwickeln der Speicherfunktion	Lukas Keinert
Ausplanen und Ausschreiben der Handlung sowie Übertragen dieser in die text.json Datei	Cora V. Stokowsky. Julie L. Haupt
Befehlseingabe	Cora V. Stokowsky
Präsentation und Vorbereitung dieser	Selina Zesewitz-Thiel, Cora V. Stokowsky, Lukas Keinert, Julie L. Haupt