

Projektgruppe

Programmiererlebnisse

Tom Bohne (7021407)
Niklas Hagengers (7022410)
Max Hahn (7020985)
Torben Landwehr (7022319)
Viet Hai Nguyen (7021478)
Marcus Rosengart (7013597)
Noah Saibel (7020621)
Niklas Schlüter (7020796)
Daniel Schötz (7022630)
Dennis Seiler (7011776)
Timm Seiler (7013248)
Mykyta Skrypnyk (7010890)
Jan Willms (7020619)

eingereicht am **21.01.2024**

Betreut von
Prof. Dr. Carsten Link
Frederik Gosewehr, M.Eng.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Verwendungszweck / Zielgruppe	1
1.3 Aufbau der Arbeit	1
2 Learn Python with Turtle Graphics	2
2.1 Produktbeschreibung	2
2.1.1 Ziel und Zweck der Anwendung	2
2.1.2 Darstellung der Webanwendung	3
2.1.3 Funktionalität der Webanwendung	3
2.1.4 Benutzergruppen	5
2.1.5 Anwendungsumgebung	5
2.1.6 Benutzerdokumentation	6
2.1.7 Annahmen und Abhängigkeiten	6
2.2 Anforderungen	7
2.2.1 Randbedingungen	7
2.2.2 Qualitätsmerkmale	7
2.2.3 Funktionale Anforderungen	7
2.3 Umsetzung	10
2.3.1 Struktur der Webanwendung	10
2.3.2 JavaScript-Bibliotheken und Python-Module im Frontend	10
2.3.2.1 brython.js	10
2.3.2.2 Turtle Graphics	11
2.3.2.3 codemirror.js	11
2.3.2.4 d3.js	11
2.3.2.5 qrcode.js	12
2.3.3 Backend	12
2.3.4 gofile.io (externer File-Hosting Dienst)	12
2.4 Schnittstellenbeschreibung	13
2.4.1 Laden eines Levels	13

2.4.2	Ausführung des Codes im Code-Editor	14
2.4.3	Konsolenausgabe in der Webanwendung	15
2.5	Evaluation	17
2.5.1	Vergleich zur Anforderungsanalyse	17
2.5.2	Ausblick	18
2.5.2.1	Erweiterbarkeit	19
2.5.2.2	Zukunftsauussicht	19
3	Programmieren mit Audio	21
3.1	Einleitung	21
3.2	Benutzer	22
3.2.1	Benutzergruppen	22
3.2.2	Dokumentation für Bentuzer	22
3.3	Anforderungen	23
3.4	Umsetzung	23
3.4.1	Auswahl der richtigen Technologie	23
3.4.2	Entwicklungsumgebung	24
3.4.2.1	Auswahl der Entwicklungsumgebung	24
3.4.2.2	Umbau der Entwicklungsumgebung	24
3.4.2.3	Installation	25
3.5	Fehler	27
3.6	Fazit und Ausblick	27
4	Dokumentation: CodeClash	28
4.1	Die Einleitung	28
4.2	Die Anforderungen an das Projekt	28
4.3	Die Entwicklungsumgebung	28
4.4	Grafische Oberfläche	28
4.4.1	Hauptmenü GUI	28
4.4.2	Level GUI	29
4.4.3	Blöcke	30
4.5	Beispiel	31
4.6	Installation für Mac-Geräte	31
4.7	Ausblick	31

5 Farbzauber und Funktionen (Vormals ASCII)	33
5.1 Produktbeschreibung	33
5.1.1 Ziel und Zweck der Anwendung	33
5.1.2 Swing-Anwendung	34
5.1.2.1 C++ Code	34
5.1.2.2 Kreative Aufgabenkonzeption	35
5.1.2.3 Erweiterung der Übung	35
5.1.2.4 Anpassbare Benutzererfahrung	35
5.1.2.5 Weitere Werkzeuge	35
5.1.3 Benutzergruppen	36
5.1.4 Anwendungsumgebung	36
5.1.5 Benutzerdokumentation	36
5.1.6 Annahmen und Abhängigkeiten	36
5.2 Anforderungen	37
5.2.1 Rahmenbedingungen	37
5.2.2 Funktionale Anforderungen	37
5.3 Umsetzung	39
5.3.1 Triangle/Circle/Rectangle Fill	39
5.3.2 Zeichnen	39
5.3.3 Developer View	40
5.3.3.1 Developer View	40
5.3.4 C++ zu Java	40
5.4 Evaluation	41
5.4.1 Vergleich zu den Anforderungen	41
6 Programmieren lernen mit dem MIT App Inventor	43
6.1 Produktbeschreibung	43
6.1.1 Der MIT App Inventor	43
6.1.1.1 Funktionen	43
6.1.1.2 Bedienung	44
6.1.2 Aufgaben	44
6.2 Anforderungen	45
6.2.1 Rahmenbedingungen	45

6.2.2	Aufgabe	45
6.2.3	Dokument für Schüler	45
6.2.4	Dokument für Lehrende	45
6.3	Umsetzung	46
6.3.1	Schwierigkeiten	46
6.3.2	Art der zu Programmierenden Anwendung	46
6.3.3	Aufbau der To-Do Aufgabe	46
6.4	Resultat	47
6.4.1	Zukunft	47
7	Fazit	48
7.1	Zusammenfassung	48
7.2	Gewonnene Erkenntnisse	48
Literaturverzeichnis		i
Abbildungsverzeichnis		ii
Listings		iii
A Anhang - Learn Python with Turtle Graphics		v
A.1	Ablaufdiagramme zu Schnittstellen	v
A.2	Lehrer-PDF	vii
A.3	Schüler-PDF	xvi
B Anhang - Programmieren mit Audio		xxxiii
B.1	Lehrer-PDF	xxxiii
B.1.1	Installation	xxxiii
B.1.2	Musterlösungen	xxxiv
B.1.2.1	Musterlösung Aufgabe 0 - Variablen, Arithmetik und das Ausgeben auf der Konsole	xxxiv
B.1.2.2	Musterlösung Aufgabe 1 - If/Else Verzweigung	xxxv
B.1.2.3	Musterlösung Aufgabe 2 - Schleifen	xxxvi
B.1.2.4	Musterlösung Aufgabe 3 - Töne abspielen	xxxvi
B.1.2.5	Musterlösung Aufgabe 4 - C-Dur-Tonleiter erstellen	xxxvii

B.1.2.6 Musterlösung Aufgabe 5 - ABC Notation	xxxviii
B.2 Schüler-PDF	xxxix
B.2.1 Lernziele	xxxix
B.2.2 Variablen	xl
B.2.2.1 Rechnen mit Variablen	xl
B.2.2.2 Ausgeben von Variablen	xl
B.2.3 Schleifen	xli
B.2.4 Arrays	xlii
B.2.5 If/Else	xliii
B.2.6 Funktionen	xliv
B.2.7 Musik und Töne	xlv
B.2.7.1 Die tone() Funktion	xlv
B.2.7.2 Die tones() Funktion	xlv
B.2.7.3 Die play() Funktion	xlvi
B.2.8 Aufgaben	xlvi
B.2.8.1 Aufgabe 0 - Variablen, Arithmetik und das Ausgeben auf der Konsole	xlvii
B.2.8.2 Aufgabe 1 - If/Else Verzweigung	xlvii
B.2.8.3 Aufgabe 2 - Schleifen	xlvii
B.2.8.4 Aufgabe 3 - Töne abspielen	xlvii
B.2.8.5 Aufgabe 4 - C-Dur-Tonleiter erstellen	xlviii
B.2.8.6 Aufgabe 5 - ABC Notation	xlviii
C Anhang - CodeClash	xlix
C.1 Lehrer-PDF	xlix
C.1.1 Inhaltsverzeichnis	xlix
C.1.2 Installation für Mac-Geräte	xlix
C.1.3 Lernziele	xlix
C.1.4 Benutzeroberfläche:	xlix
C.1.5 Blöcke:	l
C.1.6 Aufgabenbeschreibung	l
C.1.7 Lösungen	li
C.1.8 Level 1:	li

C.1.9 Level 2:	li
C.1.10 Level 3:	li
C.2 Schülerhandbuch: CodeClash	liv
C.2.1 Ziele:	liv
C.2.2 Einführung:	liv
C.2.3 Benutzeroberfläche:	liv
C.2.4 Blöcke:	liv
C.2.5 Zusammenfassung:	lv
D Anhang - Farbzauber und Funktionen	lvi
D.1 Anhang - SchülerPDF	lvi
D.2 Anhang - LehrerPDF	lxiv
D.3 Anhang - Online Auftritt	lxv
D.4 Anhang - Lizenz	lxviii
E Anhang - Programmieren lernen mit dem MIT App Inventor	lxix
E.1 Schüler PDF	lxix
E.1.1 Einleitung	lxix
E.1.2 Lernziele:	lxix
E.1.3 Vorbereitung	lxix
E.1.3.1 Neues Projekt erstellen	lxix
E.1.3.2 Verbinde dich mit deinem Smartphone	lxxx
E.1.4 Aufgabe 0 - Benutzeroberfläche kennenlernen (ca. 5 Minuten)	lxxxi
E.1.5 Aufgabe 1 - Bedingungen (ca. 10 Minuten)	lxxxi
E.1.6 Aufgabe 2 - Listen erstellen und anzeigen (ca. 10 Minuten)	lxxxiii
E.1.7 Aufgabe 3 - Einträge aus Liste auswählen/löschen (ca. 10 Minuten)	lxxxv
E.1.8 Aufgabe 4 - Listen permanent speichern (TinyDatenBank) (ca. 10 Minuten)	lxxxviii
E.1.9 Abschlussaufgabe - To Do List App (ca. 30 Minuten)	xc
E.2 Lehrer PDF	xciv
E.2.1 Installationsanleitung	xciv
E.2.1.1 Manuelle Installation	xciv

E.2.1.2	Installation mit Vagrant (Virtualbox)	xciv
E.2.2	Musterlösungen	xcv
E.2.2.1	Musterlösung zu 1.2	xcv
E.2.2.2	Musterlösung zu 2.2	xcv
E.2.2.3	Musterlösung zu 3.2	xcvi
E.2.2.4	Musterlösung zu 4.2	xcvi
E.2.2.5	Musterlösung zu 5.2	xcvi

1 Einleitung

Dieses Dokument geht auf die Arbeit der Projektgruppe Programmiererlebnisse unter der Leitung von Prof. Dr. Carsten Link und Frederik Gosewehr, M.Eng. ein. Ziel der Projektgruppe war die Erstellung von Aufgaben, mittels verschiedener Technologien, die es Schülern und Erstsemestern ermöglichen sollte, einen Einblick in die Programmierung zu bekommen. Die Projektgruppe wurde dabei in einzelne Arbeitspakete, sogenannten Szenarienpakete, unterteilt.

1.1 Motivation

Bei der steigenden Nachfrage an besetzten IT-Positionen und unserer eigenen Leidenschaft an der Informatik, möchten wir mit Hilfe dieser Projektgruppe Jugendliche und junge Erwachsene einen Einblick in die Welt des Programmierens geben. Somit können einerseits mehr IT-Positionen besetzt werden und andererseits in den jungen Leuten eine Leidenschaft für das Programmieren bzw. für die Informatik allgemein geweckt werden, von der sie unter Umständen vorher nicht wussten. Auch falls im nach hinein nicht in der Informatik gearbeitet wird, so ist ein besseres Verständnis für die Informationstechnik, in der heutigen Welt wo Internet of Things (IoT) in vielen Teilen des Lebens Einzug findet, sicherlich von Vorteil. Außerdem zielen die Projekte noch darauf ab dem Benutzer das Gefühl zu vermitteln das er 'Fürs Leben gelernt' hat.

1.2 Verwendungszweck / Zielgruppe

Die hier erarbeiteten Teilprojekte sollen für Veranstaltungen mit Schülern und Erstsemestern einen ersten Einblick und Umgang mit der Programmierung geben.

1.3 Aufbau der Arbeit

Wie erwähnt besteht das Projekt aus 5 Szenarienpaketen. Jedes Szenario wird in einem separaten Abschnitt erläutert. Im folgenden Abschnitt 2 wird das Szenario **Learn Python with Turtle Graphics** und die in diesem entwickelte Webanwendung mit interaktiven Python-Code-Editor beschrieben. Abschnitt 3 Programmieren mit Audio beschreibt eine interaktive Webanwendung mit Audioausgabe. In Abschnitt 4 wird ein in der Godot-Engine entwickeltes 2D-Spiel namens **CodeClash** beschrieben. In Abschnitt 5 **Farbzucker und Funktionen** geht es um eine Java Swing Anwendung die C++ Code ausliest und Formen in Farbe darstellt. Abschnitt 6 namens **MIT App Inventor** vermittelt Programmierkonzepte mittels Blocksprache.

2 Learn Python with Turtle Graphics

Dieser Abschnitt führt in das Szenario und Produkt "Learn Python with Turtle Graphics" ein. Dabei handelt es sich um eine interaktive Webanwendung, die Schülerinnen und Schülern ein Programmiererlebnis bietet. Die Webanwendung vermittelt spielerisch durch mehrere Level grundlegende Konzepte der Programmierung mit Python. Die clientseitige Logik basiert auf Brython, während Turtle Graphics für die Visualisierung im interaktiven Lernprozess verwendet wird.

Die nachfolgenden Abschnitte bieten eine detaillierte Produktbeschreibung sowie die Definition von Anforderungen mittels Anforderungsanalyse. Die Umsetzung der Programmierung und die für eine mögliche Weiterentwicklung relevanten Schnittstellen werden ebenfalls ausführlich erläutert. Abschließend erfolgt ein Vergleich des resultierenden Produkts mit den definierten Anforderungen, um den Erfolg des Projekts zu evaluieren. Ein Ausblick auf zukünftige Erweiterungsmöglichkeiten rundet diesen Abschnitt ab.

Der Anhang A enthält unterstützende Materialien, darunter eine Anleitung für die Schülerinnen und Schüler (Schüler-PDF) mit Bedienungsanleitung zur Webanwendung und Erläuterungen zu den Programmierkonzepten und zum Einsatz von Turtle Graphics. Eine Anleitung für Dozentinnen und Dozenten (Lehrer-PDF) beinhaltet Anleitungen zur Installation und Inbetriebnahme der Webanwendung, unterstützt beim Erstellen neuer Levels und gibt Anleitung zur Passwortänderung für Lösungen in der Webanwendung. Zusätzlich enthält die Lehrer-PDF Musterlösungen für jedes Level und Hilfestellungen bei typischerweise auftretenden Fehlermeldungen.

2.1 Produktbeschreibung

In diesem Abschnitt wird das Produkt beschrieben, dabei wird auf die Ziele und Zwecke der Anwendung, sowie auf Funktionalitäten und Abhängigkeiten eingegangen. Zudem wird eine Zielgruppen definiert, für die die Anwendung entwickelt wurde, welche in diesem [GitHub-Repository](#) zu finden ist. Des Weiteren wird auf zusätzlich auszuliefernde Unterlagen zur Unterstützung von Nutzern eingegangen, die unter folgenden Hyperlinks ([Schüler-PDF](#), [Lehrer-PDF](#)) auf GitHub, sowie im Anhang der Dokumentation zu finden sind. Die Schüler- und Lehrer-PDFs auf GitHub sind bei einer Ausgabe an Nutzer der im Anhang vorzuziehen, da diese über ein Inhaltsverzeichnis und arabischen Seitenzahlen verfügen, wie auch als separate Datei einfacher herauszugeben oder zu drucken sind.

2.1.1 Ziel und Zweck der Anwendung

"Learn Python with Turtle Graphics" ist eine Webanwendung, die darauf abzielt, Nutzern das Programmieren auf anschauliche Weise zu vermitteln. In

dieser Anwendung lernen Anwender durch eine Reihe von Levels die Grundlagen der Programmiersprache Python, unterstützt durch die visuellen Elemente von Turtle Graphics. Die Anwendung richtet sich an eine junge Zielgruppe im Alter von 10 bis 18 Jahren, wobei der Fokus auf einer interaktiven Lernerfahrung liegt.

2.1.2 Darstellung der Webanwendung

Die Benutzeroberfläche von "Learn Python with Turtle Graphics", zu sehen in [Abbildung 3](#), besteht aus einem Aufgabenbereich (markiert in Rot), einem intuitiven Code-Editor (markiert in Grün), einer Konsole für Ausgaben und Fehlermeldungen (markiert in Blau) und einem Canvas-Bereich (markiert in Orange), in dem die Turtle Graphics angezeigt wird.

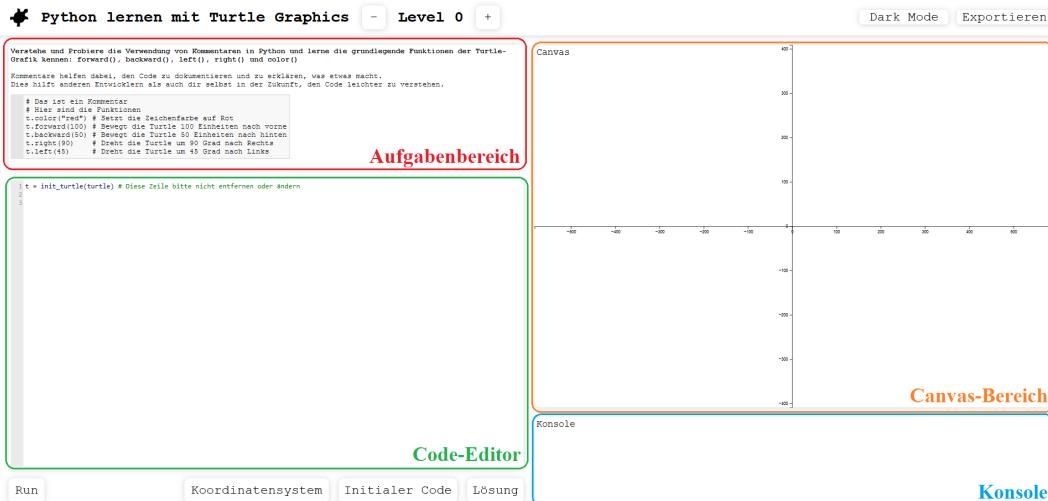


Abbildung 1: Darstellung der Webanwendung

Die Anwendung der [Abbildung 3](#) verfügt über eine benutzerfreundliche Navigation, um durch verschiedene Level zu navigieren, die mit Hilfe des "+"- und "-"Buttons in der Navigationsleiste oberhalb des Aufgabenbereiches ausgewählt werden können. Unterhalb des Code-Editors befinden sich ein Button zum Ausführen des geschriebenen Codes, ein weiterer zum Einblenden eines Koordinatensystems für bessere Orientierung, sowie Optionen zum Einsetzen des Initialen Codes und der Lösung für jedes Level. Oberhalb des Canvas-Bereichs in der Navigationsleiste befinden sich Buttons für den Wechsel zwischen Dark- und Light-Mode, sowie Optionen zum Exportieren der Level und Lösungen.

2.1.3 Funktionalität der Webanwendung

Die Webanwendung bietet viele verschiedene Funktionalitäten, die den Benutzern eine bessere und intuitiveres Lernerlebnis ermöglichen.

Interaktiver Python-Code-Editor

Der interaktive Python-Code-Editor ist eine der Schlüsselfunktionen der Webanwendung 'Learn Python with Turtle Graphics', die in Abbildung 3 zu sehen ist. Dieser ermöglicht es Benutzern, Programmierkonzepte effektiv zu erlernen, indem sie direkt Code eingeben und dessen Ausführung sofort beobachten können. Der Editor ist nicht nur mit Funktionen wie Code-Highlighting und automatischer Einrückung ausgestattet, sondern bietet auch über eine integrierte Konsole Fehlererkennung. Diese Features erleichtern das Verständnis des Codes und helfen den Benutzern, Fehler schnell zu finden und zu korrigieren.

Progressive Levelkonzeption

Die Strukturierung der Lerninhalte folgt einem progressiven Stufenkonzept, bei dem die Benutzer durch sorgfältig abgestufte Levels geführt werden. Jedes Level ist so konzipiert, dass es auf dem vorherigen aufbaut und schrittweise komplexere Aspekte der Programmierung einführt. Diese methodische Steigerung der Komplexität unterstützt die Anwender dabei, ihre Programmierkenntnisse konsequent zu erweitern und zu vertiefen.

Erweiterung der Level

Die Flexibilität der Lernumgebung in 'Learn Python with Turtle Graphics' wird durch die einfache Erweiterbarkeit der Lernstufen gewährleistet. Benutzer haben die Möglichkeit, direkt auf die Levelordner zuzugreifen, um bestehende Lernstufen zu modifizieren oder neue hinzuzufügen. Jedes Level ist als strukturiertes JSON-Objekt aufgebaut, das Anweisungen, Initial- und Lösungscode sowie spezifische Parameter und Funktionen enthält. Mehr auf die Ergänzung von Leveln wird in [Abschnitt Erstellung von neuen Levels](#) im [Anhang A.2](#) drauf eingegangen.

Visualisierung mit Turtle Graphics

Ein weiteres Merkmal ist die Visualisierung mit von Python-Code durch Turtle Graphics. Diese Funktionalität erlaubt es den Benutzern, die Auswirkungen ihrer Programmierung nach Ausführung sofort zu sehen, wodurch ein tieferes Verständnis für Code-Struktur und -Logik gefördert wird.

Anpassbare Benutzererfahrung für optimales Lernen

Die Webanwendung 'Learn Python with Turtle Graphics' bietet eine maßgeschneiderte Lernumgebung, die auf die individuellen Bedürfnisse der Benutzer zugeschnitten ist. Ein wesentliches Merkmal ist die Möglichkeit für den Benutzer, zwischen einem hellen und einem dunklen Modus zu wählen, was eine personalisierte Anzeige ermöglicht, die den visuellen Komfort verbessert

und die Ermüdung der Augen über längere Lernsitzungen hinweg reduziert. Diese Anpassungsfähigkeit trägt dazu bei, eine angenehme und ergonomische Lernatmosphäre zu schaffen, in der Benutzer unabhängig von ihren Vorlieben für Lichtverhältnisse oder Bildschirmeinstellungen optimal arbeiten können. Die einfache Umschaltfunktion zwischen den Modi fördert die Flexibilität und Zugänglichkeit der Anwendung und zeigt das Engagement für eine Benutzererfahrung, die sowohl die kognitiven als auch die physischen Aspekte des Lernens berücksichtigt.

Unterstützende Lernwerkzeuge

Zur Hilfe des Lernprozesses werden den Benutzern verschiedene Hilfswerkzeuge angeboten. Dazu gehören eine integrierte Konsole zur Ausgabe von Ergebnissen und Fehlermeldungen, sowie ein Koordinatensystem für die bessere Übersicht der Turtle Graphics und Tipps und Lösungen von Programmieraufgaben.

Exportfunktionen

Die Anwendung ermöglicht es Benutzern, ihre Arbeit zu exportieren und zu teilen. Eine Exportfunktion erlaubt das Speichern und Teilen von Code und des Canvas und das Drucken dieser, während die Integration von QR-Code-Generierung es einfach macht, Links zu Projekten oder Ergebnissen schnell und unkompliziert zu teilen.

2.1.4 Benutzergruppen

Die Hauptbenutzergruppen der Anwendung sind Schüler im Alter von **13 bis 18 Jahren**, die Grundlagen der Programmierung erlernen möchten. Die Benutzeroberfläche und die Lerninhalte sind so gestaltet, dass sie für Anfänger ohne Vorkenntnisse leicht verständlich sind.

2.1.5 Anwendungsumgebung

Die Webanwendung ist für moderne Webbrowser wie **Chrome**, **Firefox** und **Safari** optimiert. Die Anwendung ist für den Betrieb auf Endgeräten mit dem Betriebssystem **macOS** vorgesehen, ist aber auf anderen Betriebssystemen vollständig funktionsfähig. Sie erfordert für manche Features eine stabile Internetverbindung, sowie eine Vorinstallation von Python3 ist nötig. Die Anwendung wurde für niedrig auflösenden bis hin zu hochauflösenden Bildschirmen optimiert, sodass die Anwendung selbst bei kleinen Bildschirmen eine angenehme Darstellung hat.

2.1.6 Benutzerdokumentation

Zur Unterstützung der Benutzer wird eine Schüler-PDF bereitgestellt, die mehr auf die Grundlagen der Programmierung hineingeht sowie auf Tutorials im Internet hinweist.

2.1.7 Annahmen und Abhangigkeiten

Bei der Entwicklung wurde angenommen, dass die Endgerate der Benutzer bereits uber Python 3 verfugen, was die Hauptsystemanforderung darstellt. Daruber hinaus wird davon ausgegangen, dass Benutzer uber grundlegende Computer- und Browserkenntnisse verfugen und mit den bereitgestellten Tools und Funktionen umzugehen zu wissen, aber moglicherweise keine Erfahrung mit Programmierung haben.

Die Webanwendung ist abhangig von bestimmten externen Bibliotheken und Python-Modulen, die fur die Kernfunktionalitaten unerlasslich sind. **Brython.js** spielt dabei eine Schlesselrolle, da es die Ausfuhrung von Python-Code im Browser ermoglicht, was ein zentraler Aspekt der Anwendung ist. Die visuelle Darstellung des Python-Codes wird durch **Turtle Graphics** realisiert, und ist entscheidend fur das interaktive Lernerlebnis. Fur den Code-Editor ist **CodeMirror.js** von wesentlicher Bedeutung, da es Funktionen wie Syntax-Highlighting und Code-Bearbeitung bereitstellt. Die Darstellung des Koordinatensystems wird durch **d3.js** ermoglicht. Zudem spielt **qrcode.js** eine wichtige Rolle fur die Exportierungs-Funktionalitat der Anwendung, indem es die Generierung von QR-Codes fur den Downloadlink ermoglicht. Zuletzt ist die Anwendung auf den Dienst **gofile.io** angewiesen, um die Speicherung und den Zugriff auf Dateien fur den Export von QR-Codes zu ermoglichen.

In [Abschnitt 2.3.2](#) wird mehr auf die JavaScript-Bibliotheken und Python-Module eingegangen.

2.2 Anforderungen

In diesem Abschnitt wird auf die geplanten Anforderungen eingegangen. Zu diesen zählen die Randbedingungen, funktionalen Anforderungen, sowie die Qualitätsmerkmale.

2.2.1 Randbedingungen

Als erstes wurden die Randbedingungen für die Webanwendung festgelegt. Dabei handelt es sich um Voraussetzungen und Einschränkungen, die bei der Entwicklung berücksichtigt wurden.

Randbedingung 1 - Kompatibilität auf allen gängigen Browsern

- Webanwendung muss auf den neuesten Versionen der gängigen Browser wie Chrome, Firefox, Safari und Edge fehlerfrei funktionieren

Randbedingung 2 - Funktionstüchtig auf leistungsschwachen PCs

- Webanwendung muss auch auf leistungsschwachen PCs funktionieren, um eine akzeptable Nutzung sicherzustellen
- Ladezeiten der Anwendung auf leistungsschwachen PCs sollten akzeptabel sein

2.2.2 Qualitätsmerkmale

Bei den Qualitätsmerkmalen werden Merkmale definiert, die wichtige Kriterien für die Umsetzung der Webanwendung sind.

Qualitätsmerkmal 1 - Hinzufügen neuer Level

- Möglichkeit, neue Umgebungen oder Szenarien hinzuzufügen, sollte einfach und ohne erheblichen zusätzlichen Entwicklungsaufwand möglich sein.

Qualitätsmerkmal 2 - Wartbarkeit

- Gut strukturierter, dokumentierter und leicht verständlicher Code soll implementiert werden

2.2.3 Funktionale Anforderungen

In diesem Abschnitt geht es um die wesentlichen Funktionen der Webanwendung, die eine Orientierung für die Implementierung darstellen.

Anforderung 1 - Code-Editor

- Eingeben von Python-Code
- Logik-Highlighting für die Logik in verschiedene Farben (**Kann-Ziel**)

Use-Case: Der Benutzer ist in der Lage Python Code einzugeben und die Logik wird durch das Highlighting hervorgehoben.

Anforderung 2 - Visualisierung der Ergebnisse

- Ausführen des Codes durch Klick auf Button zeichnet, im Code-Editor definierte Formen auf freie Fläche
- Debugging des Codes durch Klicken auf Button lässt Zeilen einzeln ausführen (**Kann-Ziel**)

Use-Case: Der Benutzer führt den Code aus und überprüft die visuellen Ergebnisse.

Anforderung 3 - Levelwahl / LevelUp

- freie Levelwahl
- LevelUp mit Überprüfung der Programmierlogik (**Soll-Ziel**)
- LevelUp nach Wissensabfrage zu dem im Level erlernten Programmierkonzept (**Soll-Ziel**)
 - sollte überspringbar sein

Use-Case: Der Schüler wählt frei ein neues Level.

Anforderung 4 - Aufgabenstellung

- Darstellung der Aufgabe als Text
- Darstellung von Hinweisen als Code
- Musterlösung der Aufgabe (**Soll-Ziel**)
 - Passwortgeschützt oder versteckter Button

Use-Case: Der Schüler sieht Hinweise und Musterlösung nach erfolgreicher Authentifizierung und schließt mithilfe dieser das Level erfolgreich ab.

Anforderung 5 - Handout für Schüler

- Drucken des Codes und der bemalten grafischen Oberfläche (**Soll-Ziel**)
- Hochladen des Codes und der bemalten grafischen Oberfläche auf File-Hosting-System. Freigeben an Schüler als Download-Link in Form von QR-Code (**Soll-Ziel**)
- Senden des Codes und bemalten grafischen Oberfläche per E-Mail (**Kann-Ziel**)

Use-Case: Ein Benutzer möchte den Code mit nach Hause nehmen und kann dies über eine der oben genannten Methoden tun.

2.3 Umsetzung

Dieser Abschnitt erläutert die Umsetzung der Webanwendung. Hierbei wird die Struktur der Webanwendung, Auswahl der eingesetzten Tools, Servertechnologien, Bibliotheken und externe Dienste begründet.

2.3.1 Struktur der Webanwendung

Die Struktur der Webanwendung wie auch die Interaktionen zwischen Komponenten dieser ist in [Abbildung 2](#) dargestellt. Das Target-Device bezeichnet das Gerät, auf dem die Webanwendung ausgeführt wird, während User Smartphone das Smartphone des Nutzers der Webanwendung ist.

Der HTTP-Server stellt dem Frontend die erforderlichen Dateien bereit. Das Frontend greift lesend auf das Dateisystem zu, um Levels zu laden.

Außerdem erfolgen Interaktionen mit dem File-Hosting-Dienst gofile.io, der in [Abschnitt 2.3.4](#) näher erläutert wird. Auf dem Frontend kann der Nutzer Dateien auf diesen Dienst hochladen. Anschließend kann er die Dateien auf seinem Smartphone mit den Browser auf der Download-Seite des Dienstes herunterladen.

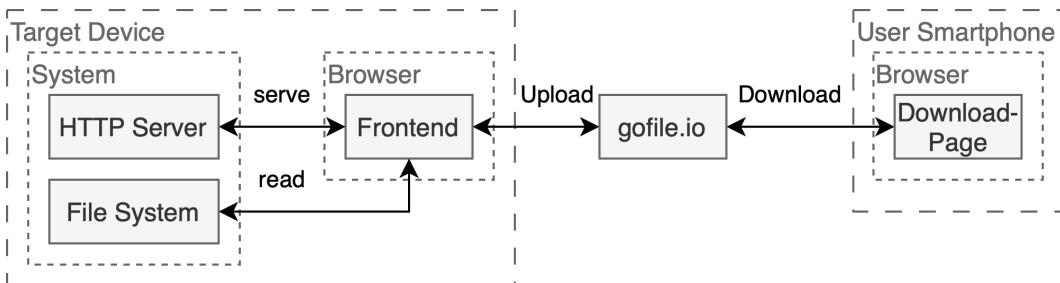


Abbildung 2: Struktur der Webanwendung

2.3.2 JavaScript-Bibliotheken und Python-Module im Frontend

Die Implementierung des Frontends der Webanwendung erfolgte unter Verwendung verschiedener JavaScript-Bibliotheken und Python-Module. Der nachfolgende Abschnitt erläutert die Gründe für ihren Einsatz sowie die spezifischen Anwendungsbereiche.

2.3.2.1 brython.js

Für die Umsetzung der clientseitigen Logik der Webanwendung kam brython.js zum Einsatz. Diese JavaScript-Bibliothek ermöglicht die Verwendung von Python anstelle von JavaScript für die clientseitige Logik.

Die Entscheidung für brython.js wurde maßgeblich durch die Anforderung getroffen, Python-Code im Browser auszuführen. Durch die Verwendung von Brython ist ein aufwendigeres Replizieren der Logik des Python-Editors in JavaScript nicht notwendig.

Diese Herangehensweise erwies sich als vorteilhaft, da sie die nahtlose Integration von Python in den Webbrower ermöglichte, insbesondere in einem Python-spezifischen Anwendungsfall wie einem Python-Editor.[1]

2.3.2.2 Turtle Graphics

Die Turtle Graphics wird verwendet, um den im Code-Editor verfassten Python-Code auf einer Zeichenfläche zu visualisieren. Gemäß den Anforderungen wird durch einen Klick auf einen Button die Zeichnung der im Code definierten Formen auf einer freien Fläche ausgelöst.

Das Visualisierungstool Turtle Graphics zu verwenden, wurde aufgrund der Unterstützung einer im Funktionsumfang reduzierten Turtle-Variante für den Browser durch brython.js beeinflusst. Weitere entscheidende Faktoren sind die weitreichende Verbreitung, hohe Akzeptanz und die Unterstützung durch die Community.[5]

2.3.2.3 codemirror.js

CodeMirror wurde für die Entwicklung des Code-Editors in der Webanwendung verwendet. Diese Bibliothek bietet umfangreiche Funktionen, darunter Code Highlighting, welches die Darstellung von Schlüsselwörtern verbessert und den Verständnis des Codes erleichtert. Durch die Möglichkeit, Python als Programmiersprache zu wählen, ist eine einfacheres, spezifischeres Syntaxhighlighting möglich. Zusätzlich bietet CodeMirror Hotkeys zur Vereinfachung der Bedienung und Zeilenzahlen für einen besseren Überblick. Ein wesentliches Merkmal der Implementierung ist die Erstellung individueller CodeMirror-Instanzen für jedes Level, die es ermöglichen, für jedes Level eine eigene Rückgängigmachen-Verlauf (STRG+Z) zu führen. Diese Funktionalität trägt wesentlich zur Benutzerfreundlichkeit und Flexibilität des Editors bei. [2]

2.3.2.4 d3.js

Die JavaScript-Bibliothek d3.js wird für die Visualisierung eines Koordinatensystems im Canvas-Bereich der Webanwendung eingesetzt. Die Bibliothek ermöglicht eine Dynamische Anpassung der Länge der 'x'- und 'y'-Koordinate an das Canvas-Element. Besonders vorteilhaft ist dabei auf Änderungen der Ansichtsgröße, wie beispielsweise beim Zoomen in der Webanwendung, zu reagieren. Dadurch wird gewährleistet, dass das Koordinatensystem stets korrekt skaliert und präzise dargestellt wird.[3]

2.3.2.5 qrcode.js

Die Integration von qrcode.js ermöglicht die Generierung eines QR-Codes aus einem gegebenen Text. In der Webanwendung wird qrcode.js dafür verwendet, einen Download-Link als QR-Code darzustellen. Dies ermöglicht es Benutzern, den Download-Link auf dem Smartphone mithilfe der Kamera zu öffnen.[6]

2.3.3 Backend

Anstelle eines dedizierten Servers wurde der Standard Python HTTP-Server eingesetzt. Diese Entscheidung ermöglicht eine Einheitlichkeit zwischen Frontend und Backend. Zudem wird keine serverseitige Logik benötigt. Ein weiterer Vorteil besteht in der weit verbreiteten Nutzung von Python, insbesondere auf den Endgeräten der Kunden, die hauptsächlich für den Einsatz der Webanwendung verwendet werden. Diese Geräte verfügen bereits über Python 3, was neben einem Browser die einzige Anforderung an die Anwendungsumgebung darstellt.

2.3.4 gofile.io (externer File-Hosting Dienst)

In der Webanwendung wird der externe Dienst gofile.io für File-Hosting verwendet. gofile.io ermöglicht das Hochladen von Dateien und Generieren eines Download-Links. [7]

Gemäß der funktionalen Anforderungen an die Webanwendung ist eine Exportfunktion implementiert, die Ergebnisse des Nutzers auf einen File-Hosting Dienst hochlädt und einen QR-Code aus dem generierten Download-Link zum Download bereitstellt.

Nach erfolgreichem Export einer HTML-Datei wird diese auf gofile.io hochgeladen. Der generierte Download-Link wird, mittels der bereits erläuterten Bibliothek qrcode.js, als QR-Code dargestellt.

2.4 Schnittstellenbeschreibung

In diesem Abschnitt werden relevante Schnittstellen für die Weiterentwicklung erläutert. Hierzu gehören das Laden von Levels, die Ausführung von Code im Code-Editor und die Konsolenausgabe innerhalb der Webanwendung.

2.4.1 Laden eines Levels

Abbildung A.1.1 im Anhang A.1 zeigt das Ablaufdiagramm der Funktion `load_level()`. Die Funktion ist in dem module `level_handler.py` definiert und in Listing 1 dargestellt ist. Diese wird zum Laden eines Levels ausgeführt. Es ist zu erkennen, dass das Laden eines Levels sowohl beim initialen Starten und Neuladen der Webanwendung als auch beim Auswählen eines Levels über die Schaltflächen mit Plus- und Minus-Symbol, die wiederum die Funktionen `previous_level()` und `next_level()` auslösen, erfolgt.

```
Listing 1: load_level()
1 def load_level():
2     canvas.clear_canvas()
3     console.clear_console()
4
5     level_parameter = read_level_from_json_file(level_index)
6
7     set_level_title(level_index)
8
9     set_tutorial(level_parameter["tutorial"])
10
11    if is_already_loaded(level_index):
12        code_mirror.show_editor(level_index)
13    elif "init_code" in level_parameter:
14        code_mirror.create_code_editor(level_index,
15            level_parameter["init_code"])
16        already_loaded[level_index] = 1
17
18    initcode.set_initcode(level_parameter["init_code"])
19
20    solution.set_solution(level_parameter["solution_code"])
21
22    theme.set_highlighting_theme()
```

Zunächst wird das zu ladende Level ermittelt und überprüft, ob dieses existiert. Anschließend wird mit der Funktion `read_level_from_json_file()`, welche in Listing 2 dargestellt ist, das Level aus dem Dateisystem im Ordner "levels" geladen. Dabei handelt es sich um eine JSON-Datei mit der Bezeichnung "level_n", wobei "n" für die Nummer des Levels steht.

```
Listing 2: read_level_from_json_file()
1 def read_level_from_json_file(level_index):
2     json_file_path = f"/levels/level_{level_index}.json"
```

```

3     with open(json_file_path, "r") as json_file:
4         level_parameter = json.load(json_file)
5
6     return level_parameter
7

```

Die Struktur der JSON-Datei eines Levels wird in [Abschnitt Level Struktur](#) in [Anhang A.2](#) erläutert. Die in der JSON-Datei definierten Strings `tutorial`, `init_code` und `solution_code` werden für folgende Aktionen verwendet:

- Falls das Level noch nicht geladen wurde, wird `init_code` in den Code-Editor eingefügt.
- `tutorial` wird im Tutorial-Bereich angezeigt.
- `init_code` wird im Modal "Initialer Code" hinterlegt.
- Das Lösungs-Modal mit der Lösung in zwei Schritten wird mit `solution_code` umgesetzt.

2.4.2 Ausführung des Codes im Code-Editor

Das Ablaufdiagramm zur Ausführung des Codes im Code-Editor ist auf der [Abbildung A.1.2](#) im [Anhang A.1](#) zu erkennen. Die Ausführung findet in der Funktion `run_code()`, dargestellt in [Listing 3](#), statt. Diese Funktion wird durch das Klicken auf die Schaltfläche "Run" sowie durch das Zoomen in der Webanwendung (falls der Canvas-Bereich nicht leer ist) ausgelöst.

```

Listing 3: run_code()
1 def run_code(ev):
2     console.clear_console()
3
4     code = code_mirror.getCodeMirrorContent(level_handler.
level_index)
5     code_with_turtle_rollback = code + "\n\nturtle.done()"
6
7     turtle.restart()
8     exec(code_with_turtle_rollback)
9
10    export.edit_level_container(
11        level_handler.level_index, code_mirror.editors[
12            level_handler.level_index]
13    )

```

Vor der Ausführung wird der Code im Code-Editor in einen String extrahiert. An diesen String wird "`\n\nturtle.done()`" angehängt, um die definierte Zeichnung in Turtle Graphics auszulösen. Der Code wird dann mittels `exec()` ausgeführt.

Turtle Graphics generiert dabei ein animiertes SVG-Element mit der Zeichnung innerhalb des div-Container mit der ID `canvas`. Aus dem Grund, dass das animierte SVG-Element beim Zoomen in der Webanwendung nicht skaliert wird, muss das animierte SVG-Element im Falle eines nicht leeren Canvas-Bereichs und eines Zoomvorgangs neu generiert werden.

2.4.3 Konsolenausgabe in der Webanwendung

Das Ablaufdiagramm für die Konsolenausgabe mittels `print()` oder bei einem Error ist in [Abbildung A.1.3](#) im [Anhang A.1](#) dargestellt.

Dabei wird das normale Verhalten, bei dem Ausgaben in die Browserkonsole geschrieben werden, unterdrückt. Stattdessen erfolgt die Ausgabe in der internen Konsole der Webanwendung.

Die Konfiguration zur Weiterleiten der Ausgaben auf der Konsole ist in [Listing 4](#) die, die Funktion `redirect_prints_and_errors_to_console()` zeigt, dargestellt.

```
Listing 4: redirect_prints_and_errors_to_console()
1 def redirect_prints_and_errors_to_console():
2     sys.stdout.write = writeConsole
3     sys.stderr.write = writeConsole
```

Hierbei werden Konsolenausgaben an die Funktion `write_console()`, dargestellt in [Listing 5](#), weitergeleitet.

```
Listing 5: write_console()
1 def write_console(*args):
2     if args[0] == "\n":
3         document["console"].html += "<br/>"
4     else:
5         extracted_part =
6         extract_relevant_part_of_error_message("".join(args))
7         document["console"].html += extracted_part
```

Wenn es sich um einen Error handelt, der durch den Nutzer innerhalb des Code-Editors ausgelöst wurde, wird lediglich der für den Nutzer relevante (letzte) Teil des Stack Traces angezeigt.

Falls es sich um einen anderen Fehler oder eine Ausgabe des Nutzers mittels `print()` handelt, wird die gesamte Meldung in der Konsole angezeigt. Die geschieht in der Funktion `extract_relevant_part_of_error_message()`, dargestellt in Listing 6.

```
Listing 6: extract_relevant_part_of_error_message()  
1 def extract_relevant_part_of_error_message(error_message):  
2     separator = '<string>, '  
3  
4     if separator in error_message:  
5         extracted_message = error_message.split(separator)  
6         [1]  
7         return re.sub(r'<module>', 'Code-Editor',  
8         extracted_message)  
9     else:  
10        return error_message
```

2.5 Evaluation

Bei der Evaluation wird als erstes der Vergleich zur Anforderungsanalyse behandelt. Dabei wird verglichen, welche Anforderungen mit den entsprechenden Randbedingungen und Qualitätsmerkmalen erfüllt wurden und wo es Abweichungen gab.

Im Anschluss wird der Ausblick beschrieben. In diesem Abschnitt werden die Möglichkeiten der Erweiterung der Webanwendung thematisiert und einen Blick in die Zukunft der Software geworfen.

2.5.1 Vergleich zur Anforderungsanalyse

Im Rahmen der Evaluierung des Projekts im Vergleich zur Anforderungsanalyse zeigt sich, dass die Webanwendung erfolgreich die festgelegten Randbedingungen und Qualitätsmerkmale erfüllt wurden.

Randbedingung 1 - Kompatibilität auf allen gängigen Browsern

Die Webanwendung wurde so entwickelt, dass sie mit den gängigsten Browsern wie Chrome, Firefox, Safari und Edge fehlerfrei funktioniert.

Randbedingung 2 - Funktionstüchtig auf leistungsschwachen PCs

Die Implementierung wurde darauf ausgelegt, auch auf leistungsschwachen PCs zu funktionieren.

Qualitätsmerkmal 1 - Hinzufügen neuer Level

Die Implementierung ermöglicht es einfach und ohne erheblichen zusätzlichen Entwicklungsaufwand, neue Umgebungen oder Szenarien in die Anwendung einzufügen. Dazu wurde ein Skript geschrieben was es ermöglicht ein neues Level zu definieren und in den Code einzufügen.

Qualitätsmerkmal 2 - Wartbarkeit

Erreicht wurde dieses Qualitätsmerkmal indem der Code strukturiert und dokumentiert, sowie in verschiedene Module aufgeteilt und kommentiert wurde.

In Bezug auf die funktionalen Anforderungen wurden die meisten Ziele erreicht.

Anforderung 1 - Code-Editor

Die Anforderung, Python-Code einzugeben und die Logik durch farbliches Highlighting hervorzuheben, wurde erfolgreich umgesetzt. Der Benutzer kann Python-Code im Code-Editor eingeben, und die Logik wird durch das entsprechende Python-Highlighting visuell hervorgehoben.

Anforderung 2 - Visualisierung der Ergebnisse

Der Benutzer kann den Code ausführen und auf dem Canvas wird die entsprechende Form gezeichnet. Das Debuggen einzelner Codeteile konnte aufgrund technischer Schwierigkeiten und Zeitmangel nicht erfolgreich umgesetzt werden.

Anforderung 3 - Levelwahl / LevelUp

Die Anforderung bezüglich freier Levelwahl wurde erfolgreich realisiert. Ein Schüler kann frei zwischen den Levels wechseln und der bereits geschriebene Code wird zwischengespeichert. LevelUp mittels Überprüfen der Programmierlogik und Wissensabfrage wurde nicht umgesetzt.

Anforderung 4 - Aufgabenstellung

Die Umsetzung der Aufgabenstellung beinhaltet die Darstellung der Aufgabe als Text sowie die Anzeige von Hinweisen als Code. Zusätzlich wurde die Musterlösung als Soll-Ziel erfolgreich integriert, wobei der Zugang passwortgeschützt ist.

Anforderung 5 - Handout für Schüler

Die Anforderung, den Code das Canvas zu drucken, wurde erfolgreich umgesetzt. Ebenso ermöglicht die Webanwendung das Hochladen des Codes und des Canvas auf ein File-Hosting-System, wobei ein Download-Link in Form eines QR-Codes für die Schüler freigegeben wird. Zusätzlich wurde eine weitere Anforderung erfolgreich implementiert, die es den Schülern gestattet, die bearbeiteten Level als HTML-Datei zu exportieren. Diese Funktion ermöglicht eine übersichtliche Zusammenfassung, indem sowohl der Programmcode als auch das Canvas-Grafikelement in einem Dokument präsentiert werden.

Das Senden der exportierten HTML per E-Mail wurde aus Gründen des Aufwands nicht realisiert, da hierfür einen Mail-Provider benötigt.

2.5.2 Ausblick

In diesem Abschnitt wird ein Ausblick im Bezug auf die Erweiterbarkeit und Zukunftsaussicht gegeben.

2.5.2.1 Erweiterbarkeit

Die Anwendung ist leicht erweiterbar, wie bereits durch die Nutzung einer anderen Projektgruppe für die Musikanwendung zur Generierung von Musik über Python-Code demonstriert wurde. In diesem Kontext wurde der Code-Editor sowie die allgemeine Struktur der Anwendung verwendet und das Canvas wurde durch ein Notenblatt ersetzt. Die Modularität des Code-Editors, der Ausführungslogik und der Struktur der Webanwendung ermöglicht es, diese für weitere Anwendungszwecke einzusetzen. Dies wurde durch die Erfüllung des in der Anforderung aufgeführten zweiten Qualitätsmerkmals erreicht. Etwaige Anpassungen bezüglich der Benutzeroberfläche sind einfach umzusetzen, da für diese Elemente bereits Container existieren.

2.5.2.2 Zukunftsaussicht

Im Folgenden werden Erweiterungsmöglichkeiten für mögliche zukünftigen Projekten vorgestellt. Hierbei handelt es sich um Erweiterungen der Funktionalität der aktuellen Webanwendung, sowie Überlegungen für weitere Webanwendungen auf Basis der modular entwickelten Webanwendung.

Dynamisches Zeichnen mittels Benutzereingabe in Textfeld oder Belegen von Tasten:

Beispielsweise könnte eine Erweiterungsmöglichkeit daraus bestehen, dem Benutzer zu ermöglichen, über Benutzereingaben in einem Textfeld oder bei Binden von Funktionen an Tastatur-Tasten (beispielsweise Pfeiltasten) die Turtle zu bewegen und somit dynamisch zu zeichnen. Diese Möglichkeit wurde von den Entwicklern der Webanwendung betrachtet aber nicht umgesetzt. Dies liegt wie in [Abschnitt 2.3.2.2](#) beschrieben, daran das es sich bei der genutzten Turtle Graphics Variante um eine im Funktionsumfang reduzierte Variante handelt. Mit dieser ist es nicht möglich dynamisch auf Benutzereingaben zu reagieren.

Zeichnen auf HTML5 Canvas:

Eine weitere Idee ist es direkt auf das Canvas, mittels Kontext des Canvas `ctx = canvas.getContext('2d')`, zu zeichnen. Dies ist unabhängig von Turtle Graphics und weiteren Visualisierung-Tools möglich, da das Zeichnen auf Canvas in HTML5 eingebaut ist. Diese Funktionalität ist vom Kunden in einem späten Stadium der Entwicklung vorgeschlagen worden. Aus zeitlichen Gründen wurde diese Idee allerdings nicht weiter verfolgt.

Hilfestellungen für Nutzer:

Konsolenausgaben mit dazugehöriger textueller Erklärung könnten es Nutzer vereinfachen Fehlermeldungen eigenständig zu verstehen. Dies würde dazu führen das Nutzer Fehler eigenständig korrigieren können und somit simple Fragen zu Fehlermeldungen an Dozentinnen und Dozenten reduziert werden.

Eine Schaltfläche zum Zeichnen der Musterlösung ohne diese zu kennen, würde die Komplexität beim Erstellen eines Levels reduzieren und nach Fragen zum

zu zeichnenden Objekt eliminieren. Dadurch werden Levels vermutlich schneller erledigt. Somit lässt sich die Anzahl der Levels steigern, was mutmaßlich dazu führt das noch mehr Programmierkonzepte als bisher in der selben Zeit vermittelt werden können.

Obwohl das Debuggen von Code aufgrund technischer Schwierigkeiten und Zeitmangel bisher nicht erfolgreich umgesetzt wurde, könnte dies ein interessanter Ansatz für die Zukunft sein. Die Schüler könnten so besser verstehen, wo entsprechende Fehler sind und diese nachvollziehen.

Möglich wäre es auch den vom Nutzer geschriebenen Code mittels Cookies im Browser zu speichern, damit dieser nach Neuladen der Seite bzw. Anwendung noch vorhanden ist. Dies senkt die Frustration wenn dies unbeabsichtigt auftritt.

Features zur Kontrolle des Fortschritts von Nutzern:

Das als Soll-Ziel geplante Feature des LevelUps mittels Überprüfung der Programmierlogik oder Wissensabfrage, würde dafür sorgen das Nutzer Levels in definierter Reihenfolge abarbeiten und sich nicht mit überspringen der Levels überfordern können.

Zusätzlich wäre es sinnvoll für jedes Level ein separates einzigartiges Passwort zu definieren. Dieses könnte als weiterer String in der JSON-Datei eines jeden Levels gespeichert werden. Somit ist es möglich Passwörter zu Levels bekannt zu geben, ohne das alle Lösungen bekannt sind. Dozentinnen und Dozenten haben somit die Kontrolle über den Fortschritt. Dass dadurch die meisten Nutzer beim gleichen Level sind, hilft Dozentinnen und Dozenten dabei Hilfestellungen zu geben, da sich Fragen der Nutzer nur auf ein Level beziehen.

Weitere Anwendungsmöglichkeiten (außerhalb vom Zeichen)

Des Weiteren besteht die Möglichkeit weitere Anwendungen, wie beispielsweise die Musik-Anwendung, umzusetzen. Weitere Anwendungen könnten das Zeichnen mittels ASCII in der Konsole oder das Anlegen und Steuern von HTML-Elementen in definierten Containern sein. Selbstverständlich lassen sich einige der Erweiterungen zur Hilfestellungen für Nutzer und zur Kontrolle des Fortschritt auch bei diesen Anwendungsmöglichkeiten anwenden.

3 Programmieren mit Audio

3.1 Einleitung

Das Szenarienpaket hat das Ziel mithilfe externer Technologien und einer geeigneten IDE (Integrated Development Environment) Schülern anhand einer Reihe von Aufgaben die Grundbausteine der Programmierung näherzubringen. Besonderes Augenmerk soll darauf gelegt werden, dass Schüler das Feedback ihrer selbst geschriebenen Programme nicht nur über eine Konsolenausgabe bekommen, sondern auch über eine Audioausgabe anhand von Tönen oder Melodien.

Im folgenden werden Anforderungen an das Szenarienpaket formuliert. Es wird auf die Auswahl der richtigen Technologien sowie IDE eingegangen, der Umbau bzw. die Einrichtung der IDE erläutert sowie die Installation und auftretende Fehler beschrieben. Zum Schluss wird noch ein Ausblick gegeben, wie es mit dem Szenarienpaket in Zukunft weiter gehen könnte.

3.2 Benutzer

3.2.1 Benutzergruppen

Der Webeditor und die Aufgaben sind darauf ausgelegt, von Schülern im Alter von 13 bis 18 Jahren verwendet zu werden. Die Schüler brauchen hierbei keine Vorkenntnisse.

3.2.2 Dokumentation für Benutzer

Im Anhang B finden Sie eine Lehrer- sowie eine Schüler PDF, die den jeweiligen Nutzer unterstützen soll. Im Lehrer PDF befinden sich Anweisungen zur Installation des Webeditors sowie die Musterlösungen zu den einzelnen Aufgaben. Im Schüler PDF werden zu erst die Grundbausteine der Programmierung erklärt sowie die Schnittstellen Benutzung, um Töne und ABC-Notationen abzuspielen. Am Ende befinden sich dann die Aufgabenstellungen, die man auch im Webeditor findet.

3.3 Anforderungen

Die Anforderungen an das Szenarienpaket sind eine geeignete Technologie zu finden, die es ermöglicht, Töne, Melodien und Musik möglichst einfach abzuspielen. Das ganze soll in einer IDE geschehen, die so unkompliziert wie möglich ist und den Schülern vieles abnimmt, damit diese sich nicht durch zu viele Optionen und Einstellungen überfordert fühlen.

3.4 Umsetzung

3.4.1 Auswahl der richtigen Technologie

Die Auswahl der richtigen Technologie für die gestellten Anforderungen war nicht einfach. Das lag unter anderem daran, dass das Abspielen von Ton unter Verwendung externer Bibliotheken eine gewisse Anforderung an die Kenntnisse des Anwenders stellt.

Im folgenden werden Technologien aufgezählt, die dann aber doch wieder verworfen wurden.

Als erstes wurden sich **Orca**, **Glicol** und **Soul** angeschaut, alle drei sind mächtige Werkzeuge, mit denen man Musik und Melodien erstellen kann. Für das Szenarienpaket waren sie leider zu kompliziert, da alleine das Einarbeiten der Schüler, damit sie etwas eigenes erstellen können, sehr lange gedauert hätte.

Danach wurde sich **Blockly** und **Scratch** angeschaut, beides sind gute Technologien, um gerade einer eher jüngeren Zielgruppe Grundkenntnisse über die Programmierung beizubringen. Sie haben aber leider nicht so zum Szenarienpaket gepasst, wie das Entwicklerteam es sich vorgestellt hatte.

Dann wurde die Idee von Steve Hiehn aus seinem Artikel "[How to generate music with Python: The Basics](#)" überprüft. Hier werden verschiedene Python Bibliotheken benutzt, um MIDI Dateien zu erstellen. Das Problem ist, das man für das Abspielen eine DAW (Digital Audio Workstation) benötigt, das würde dem Szenarienpaket eine zu große extra Abhängigkeit hinzufügen.

Am Ende wurde sich für die Javascript Bibliothek **musical.js** entschieden. Diese Bibliothek bietet die Möglichkeit, einfache Töne sowie ABC-Notationen abzuspielen. Die **ABC-Notation** wird verwendet, um Musiknotationen mithilfe einer ASCII-Codierung in Textform darzustellen.

Um die Nutzung der Bibliothek weiter zu vereinfachen, wurden 4 Funktionen namens `tone()`, `tones()`, `play()`, `stop()` erstellt. Hierbei kümmert sich `tone()` um das Abspielen einfacher Töne, `tones()` um das Abspielen mehrerer Töne hintereinander, `play()` darum eine ABC-Notation, die vorher in einen String umgewandelt wurde, abzuspielen und `stop()` um die Möglichkeit Ton der gerade abgespielt wird, zu stoppen.

3.4.2 Entwicklungsumgebung

3.4.2.1 Auswahl der Entwicklungsumgebung

Um die Schüler nicht zu überfordern, sollte eine einfache und unkomplizierte IDE gewählt werden. Dies gestaltete sich schwierig, da es zwar solche IDEs gibt, sie aber entweder nur für ein Betriebssystem verfügbar sind oder unterschiedliche Installer für die unterschiedlichen Systeme benutzen. Die Umsetzung wäre also möglich gewesen, aber nur mit verschiedenen Installationswegen für die verschiedenen Betriebssysteme, das hätte die Installation verkompliziert und sollte möglichst verhindert werden.

Eine andere Möglichkeit wäre gewesen einen eigenen Webeditor zu programmieren, der auf einem lokalen Webserver läuft. Dann könnte man auf den verschiedenen Betriebssystemen einfach nur den Webserver starten und man könnte über einen installierten Browser auf den Editor zugreifen. Ein eigener Webserver hätte viel Arbeit bedeutet, jedoch machte einer der Auftraggeber darauf aufmerksam, dass die Gruppe des Szenarienpakets Learn Python with Turtle Graphics "bereits einen eigenen lokal gehosteten Webeditor programmiert hatten und man diesen benutzen könnte. Nach einem kurzen Meeting mit der Gruppe des anderen Szenarienpakets und einer gut aus gefallenen Bewertung, ob der Editor genutzt werden könnte, wurde entschieden, das es der beste und einfachste Weg wäre, den Editor zu übernehmen und für die Anforderungen dieses Szenarienpakets umzubauen. Tiefergehende Informationen über Funktion und Aufbau des Webeditors finden Sie im Kapitel 2.3.

3.4.2.2 Umbau der Entwicklungsumgebung

Ausgangspunkt des Umbaus war der Webeditor der Gruppe Learn Python with Turtle Graphics".

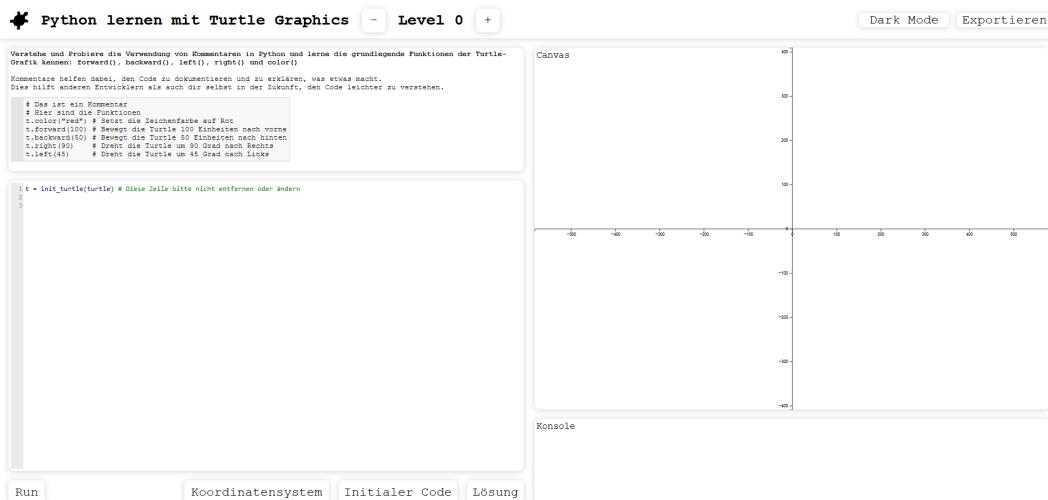


Abbildung 3: Darstellung der Webanwendung

Der Editor wurde hauptsächlich gekürzt. So wurden z. B. der Canvas und das Koordinatensystem entfernt, da es für dieses Szenarienpaket nicht gebraucht

wurde. Aus Zeitgründen wurde das Exportieren Feature komplett gestrichen. Da in nur einer Aufgabe initialer Code verwendet wird, wurde auch dieser Button entfernt. Die im Webeditor integrierten Lösungen wurden als Kapitel Musterlösungen in das Lehrer PDF ausgelagert.

Hinzugefügt wurde der stop Button, der dazu dient, den Ton nach dem Abspielen wieder zu stoppen.

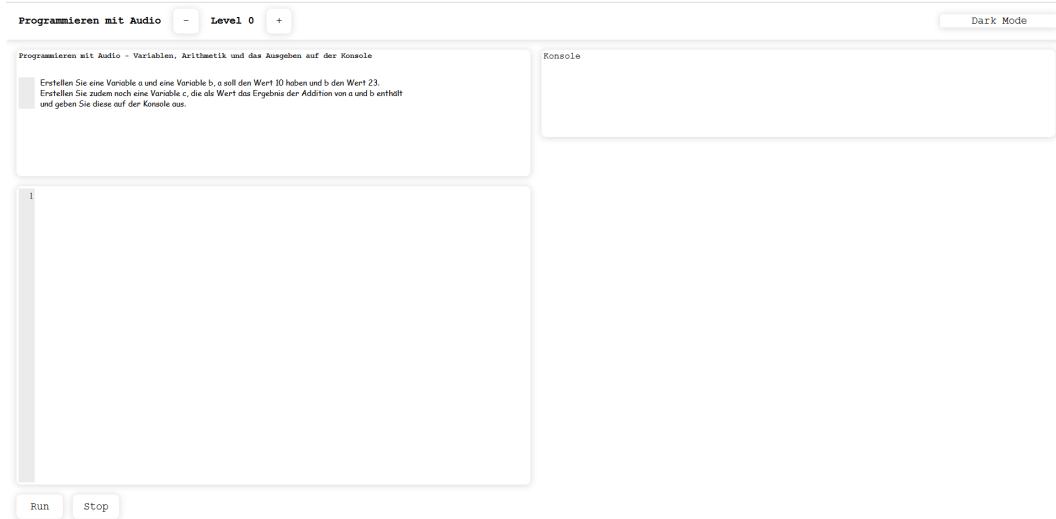


Abbildung 4: Darstellung der gekürzten Webanwendung

Am Ende benutzt dieses Szenarienpaket eine gekürzte Version des Webeditors der Learn Python with Turtle Graphics"Gruppe. Der Editor erfüllt die Anforderungen an eine unkomplizierte Entwicklungsumgebung, mit der sich einfach arbeiten lässt.

3.4.2.3 Installation

Dank des Einsatzes eines lokal gehosteten Webeditors ist die Installation mit 10 Schritten auf allen Betriebssystemen gleich. Und nur davon abhängig, dass auf dem Betriebssystem Python installiert ist.

Die Installation erfolgt in folgenden Schritten:

- Stellen Sie sicher, das auf dem Zielsystem Python installiert ist.
- Rufen Sie die Gitlab Seite der Projektgruppe auf (<https://gitlab.technik-emden.de/da6338/projektgruppe-23-link>).
- Wählen Sie den Branch `musik-soundsäus`.
- Drücken Sie auf den blauen Button `Code` und wählen unter "Download source code" das gewünschte Kompressionsformat aus. Es sollte der Download des source codes beginnen.

- Nach dem Download entpacken Sie die Datei an einen gewünschten Zielort.
- Öffnen Sie den Ordner projektgruppe-23-link-musik-sounds und folgen Sie dem Pfad "projektgruppe-23-link-musik-sounds" → "Programmieren mit Audio" → "SSource-Code". Sie sollten jetzt in einem Verzeichnis sein, das unter anderem eine HTML-Datei namens "code-editor.html" sowie ein Verzeichnis namens "levels" enthalten sollte.
- Öffnen Sie in diesem Ordner ein Terminal. Gegebenenfalls können Sie nicht direkt im Ordner ein Terminal öffnen, Sie sollten dann ein allgemeines Terminal Fenster öffnen und in den Ordner namens "SSource-Codes" navigieren.
- Sind Sie im Ordner angekommen, geben Sie `python -m http.server 8080` ein um den Server des Webeditors auf Port 8080 zu öffnen.
- Öffnen Sie jetzt einen beliebigen Browser und geben Sie in die Adressleiste `http://localhost:8080/code-editor.html`
- Es sollte sich der Webeditor des Projekts öffnen. Ab jetzt können die Aufgaben bearbeitet werden.

3.5 Fehler

Ein Fehler, der beim Testen der Aufgaben im Webeditor aufgefallen ist, ist das anstelle des Tons "H in Fehlerton ausgegeben wird. Der Fehler k nnte in der musical.js Bibliothek darin liegen, dass der Ton "Hnicht definiert ist.

3.6 Fazit und Ausblick

Das Szenarienpaket erf llt die im Kapitel Anforderungen formulierten Anforderungen an das Szenarienpaket. In Zukunft k nnte man allerdings die Aufgaben sowie den Webeditor noch deutlich erweitern. Momentan ist der Webeditor in der Anzahl der verf gbaren Aufgaben begrenzt, da es ein hartes Limit von 6 Aufgaben gibt. Dies k nnte man durch dynamisches Laden der Aufgaben  ndern und so den Lehrkr ften die M glichkeit geben, eigene weitere Aufgaben zu erstellen und hinzuzuf gen.

Zus tzlich k nnte man noch eine verworfene Idee aufgreifen und die [abc.js](#) Bibliothek benutzen, um die von den Sch lern selbst geschriebenen ABC Notationen grafisch als Notenbl tter darzustellen. Hier k nnte man dann auch das entfernte exportieren Feature wieder hinzuf gen und den Sch lern die M glichkeit geben, ihre selbst erstellen ABC Notationen als PDF oder ausgedruckt zu erhalten.

4 Dokumentation: CodeClash

4.1 Die Einleitung

Dieses Projekt hat das Ziel, ein Programm zu entwickeln, welches Schülern an einem eventuellen tag der offenen Tür, die Informatik näher zu bringen. Das Ergebnis des Projekts ist ein 2D-Spiel auf Drag&Drop basis in der Godot Engine, bei dem die Schüler lernen wie bestimmte Elemente funktionieren, die in nahezu allen Programmiersprachen vorkommen.

In der folgenden Arbeit werden die Anforderungen der Anwendung sowie die Entwicklungsumgebung genauer erläutert. Im Anschluss wird die Grafische Oberfläche der Anwendung und der grobe Ablauf eines levels beschrieben. Zum Abschluss folgt ein direktes Beispiel anhand des ersten levels, in dem nocheinmal ein konkretes Beispiel zur veranschaulichung des Spielablaufs erläutert wird.

4.2 Die Anforderungen an das Projekt

- Die Schüler sollen mithilfe dieses Spiels die grundlegenden Funktionen der Programmierung kennenlernen.
- Dabei soll das Spiel ermöglichen, dass Spieler ohne programmiererische Vorkenntnisse Programme entwerfen können.
- Der Spieler muss sich dabei keine Gedanken um die Programmiersprache, IDE oder die zugrunde liegende Syntax machen.
- Der Spieler lernt die Benutzung von Bedingungen und Schleifen zur Lösung gestellter Aufgaben.

4.3 Die Entwicklungsumgebung

Das Projekt wurde in der kostenlosen open-source game engine Godot entwickelt. Bei Godot handelt es sich um eine script-based game engine in der man 2D und 3D Szenen erstellen und mit Kontent füllen kann. Anhand einer Baumstruktur werden vorgefertigte Elemente in eine Szene gelegt. Diese Elemente können nun mit der Engine eigenen Scriptssprache GDScript implementiert werden, um z.B. einem vorgefertigten Button eine funktion zuzuweisen.

4.4 Grafische Oberfläche

4.4.1 Hauptmenü GUI

- Play-Button: Dieser Button öffnet die Levelauswahl, in der man zwischen den verschiedenen Leveln wählen kann, um diese zu spielen.



Abbildung 5: Play_button



Abbildung 6: Options_button

- Options-Button: Mit diesem Button lässt sich das Options Menü öffnen, wo sich Fullscreen und Vsync aktivieren oder deaktivieren lassen.
- Exit-Button: Wenn man das Spiel verlassen möchte, kann man diesen Button drücken.

4.4.2 Level GUI

- Leiste mit Blöcken am unteren Bildschirmrand: Diese Blöcke müssen platziert und teilweise kombiniert werden, um ein Programm zu erstellen und die Aufgabe zu bewältigen.
- Blöcke: Diese Blöcke können per Drag'n'Drop auf der “Platine” platziert werden. Sie beinhalten immer eine Funktion einer spezifischen Programmstruktur.
- Viereckige Slots auf der “Platine”: Hier werden die Blöcke platziert. Die Linien zwischen den Slots dienen danach als Verknüpfung der unterschiedlichen Blöcke.
- DELETE-Button: Mit diesem Button kann der aktuelle Levelfortschritt zurückgesetzt und es kann von vorne begonnen werden, wenn man einen Fehler gemacht hat.



Abbildung 7: Exit_button



Abbildung 8: Delete_button



Abbildung 9: Start_button

- START-Button: Sind alle Blöcke platziert, die man platzieren wollte, kann dieser Button gedrückt werden, um das Programm zu starten. Nun wird geprüft, ob das Programm funktionsfähig ist und die Aufgabe somit erfolgreich bearbeitet wurde.
- HELP-Button: Mit diesem Button lässt sich ein Pop-Up öffnen, das Hilfestellungen zur jeweiligen Aufgabe bietet.
- BACK-Button: Mit diesem Button kommt man zurück zur vorherigen Ansicht (Menüs).

4.4.3 Blöcke

- START-Block: Dieser Block ist statisch und symbolisiert den Programmstart.
- ZIEL-Block: Dieser Block ist ebenfalls statisch und symbolisiert das Ende des Programms.
- WHILE-Block: Dieser Block erhöht eine Variable (hier: x) so lange, bis der im Block angegebene Wert erreicht wurde und leitet das Signal dann zum nächsten Block weiter.



Abbildung 10: Help_button



Abbildung 11: Back_button

- IF-Block: Dieser Block prüft, ob eine Variable (hier: x) einen gegebenen Wert hat oder nicht, je nachdem wird entweder der rote (false) oder der grüne (true) Ausgang verwendet.
- END-Block: Dieser Block wird symbolisch für den nicht verwendeten Ausgang eines IF-Blocks verwendet (hier meist “false”).

4.5 Beispiel

Das Spielfeld besitzt einen unbeweglichen Start- und Zielblock, sowie Felder auf denen Blöcke platziert werden können. Der Startblock symbolisiert den Beginn des Programms und der Zielblock das Ende des Programms. Die Felder werden als Platinen dargestellt und die Verbindung der Felder sind als Leiterbahnen dargestellt. Der Spieler zieht per Drag and Drop die gewünschten Blöcke aus der unteren Leiste auf eines der Felder. Um zu prüfen, ob das Programm die gestellten Anforderungen erfüllt drückt der Spieler auf den Start-Knopf in der oberen Leiste. Sollte die Lösung des Spielers falsch sein oder der Spieler unzufrieden mit seiner Auswahl sein kann der Delete-Button gedrückt werden um das Level zurückzusetzen.

4.6 Installation für Mac-Geräte

1. Virtualbox Version 7.0.12 installieren.
2. Ubuntu Version 23.04 VM in Virtualbox aufsetzen.
3. Linux-Export des Spiels in Ubuntu herunterladen.
4. In Ubuntu das Terminal im Zielordner öffnen.
5. Das Spiel mit dem Befehl: `chmod +x CodeClash.exe.sh` ausführbar machen.
6. Mit dem Befehl `./CodeClash.exe.sh` ausführen.

4.7 Ausblick

Die grundlegenden Anforderungen, welche an das Projekt gestellt wurden, sind realisiert. Das Projektergebnis hat jedoch noch viel Potenzial. Wie der Leser sicher schon bemerkt hat gibt es derzeit nur 3 grundlegende Level, welche um weitere Level erweitert werden können. Ebenso gibt es derzeit nur simple while-schleifen und if-abfragen in den Aufgaben. Neue Blöcke wie z.B.

Addition oder Subtraktion würden dem Spiel viele neue Levelmöglichkeiten geben. Ebenso kann dynamisches Feedback in den Leveln und ein ausführliches Tutorial dem Spieler eine neue Stufe an Immersion geben. Diese Ideen und weitere könnte der nächste Student, welcher an dem Projekt arbeitet, umsetzen und erweitern.

5 Farbzauber und Funktionen (Vormals ASCII)

Dieser Abschnitt beschreibt das Szenarienpaket "Farben und Funktionen" das während der Entwicklung unter dem Namen "Ascii" geführt wurde. Dabei handelt es sich um eine interaktive Swing Anwendung und zu compilierender C++ Code, aufbauend auf der Projektarbeit von Nurullah Damla. Schülern und Erstsemestler soll so ein Programmiererlebnis geboten werden. Mittels einfacher, kleiner Schritte und durchs Experimentieren wird ein Einblick in die grundlegende Konzepte der Programmierung vermittelt. Der Nutzer arbeitet hauptsächlich mit C++, wobei aber eine Umsetzung in Python möglich ist , während eine Java Swing Anwendung für die Visualisierung im interaktiven Lernprozess verwendet wird.

Die nachfolgenden Abschnitte bieten eine detaillierte Produktbeschreibung sowie die Definition von Anforderungen. Die Umsetzung der Programmierung und die für eine mögliche Weiterentwicklung relevanten Schnittstellen werden ebenfalls ausführlich erläutert. Abschließend erfolgt ein Vergleich des resultierenden Produkts mit den definierten Anforderungen, um den Erfolg des Projekts zu evaluieren. Ein Ausblick auf zukünftige Erweiterungsmöglichkeiten rundet diesen Abschnitt ab.

Der Anhang enthält unterstützende Materialien, darunter eine Anleitung für die Schüler und Studenten mit Aufgaben (Schüler-PDF) mit Bedienungsanleitung, Erläuterungen zu den Programmierkonzepten und allgemeine Hinweise zum verbessern des Lernprozesses. Eine Anleitung für Dozentinnen und Dozenten (Lehrer-PDF) beinhaltet Anleitungen zur Installation und zwei Musterlösungen. Eine Projektseite als HTML-Dokument die für die Integration im aktuellen Labor-Webauftritt vorgesehen ist. Sowie eine GNU-Lizenzvereinbarung mit den Bedingungen und Einschränkungen unter denen das Projekt verwendet werden darf.

5.1 Produktbeschreibung

In diesem Abschnitt wird das Produkt beschrieben, dabei wird auf die Ziele und Zwecke der Anwendung, sowie auf Funktionalitäten und Abhängigkeiten eingegangen. Zudem wird eine Zielgruppen definiert, für die die Anwendung entwickelt wurde. Des Weiteren wird auf zusätzlich auszuliefernde Unterlagen zur Unterstützung von Nutzern eingegangen, welche unter anderem im Anhang der Dokumentation zu finden sind

5.1.1 Ziel und Zweck der Anwendung

"Farbzauber und Funktionen" ist eine Anwendung, die darauf abzielt, Nutzern das Programmieren auf experimentelle Weise zu vermitteln. In dieser Anwendung lernen Anwender durchs Experimentieren mit Quellcode eine Reihe von

grundlegenden Konzepten der Programmierung, unterstützt durch die visuellen Elemente einer Swing-Anwendung. Die Anwendung richtet sich an Jugendliche und Junge Erwachsene, also Personen im Alter von 13 bis 24 Jahren, wobei der Fokus auf einer interaktiven Lernerfahrung liegt.

5.1.2 Swing-Anwendung

Die Benutzeroberfläche der Swing-Anwendung, zu sehen in [Abbildung 13](#), besteht aus Knöpfen (markiert in Blau), einer Anzeige der aufgerufenen Funktionen mit Mauszeiger Position (markiert in Grün), und einem Canvas-Bereich (markiert in Orange), in dem die Grafiken angezeigt werden.

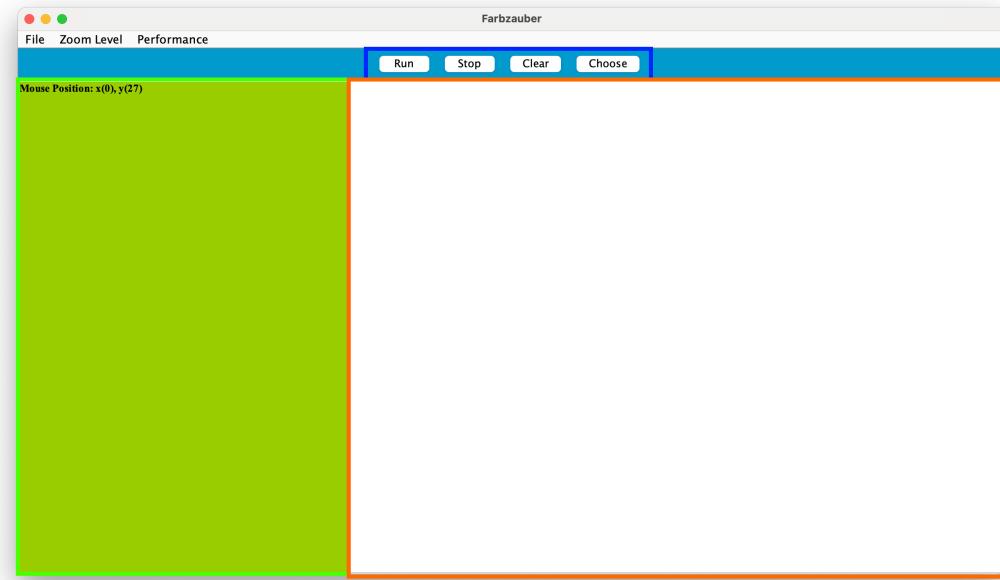


Abbildung 12: Darstellung der Swing-Anwendung

Die Anwendung der [Abbildung 13](#) verfügt über eine benutzerfreundliche und minimalistische Benutzeroberfläche, die Hauptchnittstelle erlaubt eine Datei zu laden (Choose), diese noch einmal zu zeichnen (Run), den Canvas zurückzusetzen (Clear) und das Zeichnen anzuhalten (Stop). In der oberen rechten Ecke der Anwendung finden sich weitere Optionen zum einen unter File noch einmal das Lade, erneute Zeichnen, Löschen und Stoppen, dann das Zoomen im Canvas über SZoom Level und man kann einstellen ob über den Input- bzw. Time-Counter gezeichnet werden darf.

5.1.2.1 C++ Code

Der interaktive C++ Code ist neben der Swing-Anwendung eine Schlüsselfunktionen des Teilprojektes 'Farbzauber und Funktionen'. Dieser ermöglicht es Benutzern, Programmierkonzepte experimentell und auf spielerische Weise zu erlernen und sich dabei kreativ auszuleben, indem dazu angeregt wird

mit den vorgegebenen Bausteinen (Funktionen) sein eigenes Bild zu konzipieren welches dann nach dem compilieren über die Swing-Anwendung ausgegeben werden kann.

5.1.2.2 Kreative Aufgabenkonzeption

Die Strukturierung der Lehrinhalte folgt einem kreativen Lernkonzept, bei dem die Benutzer zunächst durch Grundkonzepte der Programmierung geführt werden und dazu eingeladen sind mit diesen zu experimentieren. Jedes Abschnitt ist so konzipiert, dass es auf dem vorherigen aufbaut und schrittweise komplexere Aspekte der Programmierung einführt. Diese methodische Steigerung der Komplexität unterstützt die Anwender dabei, ihre Programmierkenntnisse konsequent zu erweitern und zu vertiefen und ihnen alle Werkzeuge zu geben die sie brauchen um Ende der Übungen ihr eigenes Bild zu erstellen.

5.1.2.3 Erweiterung der Übung

Die Flexibilität der Lernumgebung wird durch die einfache Erweiterbarkeit der Musterlösung gewährleistet. Benutzer haben die Möglichkeit, direkt ihre eigene Musterlösung zu erstellen, um zu modifizieren welche Grundkonzepte oder benötigt werden oder neue hinzuzufügen. Jede Musterlösung ist als Anregung gedacht, sie soll jeweils zeigen was theoretisch mit den gegebenen Funktionen möglich ist. Dementsprechend ist auch die Schüler-PDF gestaltet, es werden keine konkreten Anweisungen erteilt sondern lediglich Methoden und Werkzeuge dem Endbenutzer an die Hand gegeben seine eigene Kreativität auszuleben.

5.1.2.4 Anpassbare Benutzererfahrung

Das Teilprojekt bietet eine maßgeschneiderte Lernumgebung, die auf die individuellen Bedürfnisse und Fähigkeiten der Benutzer zugeschnitten ist. Ein wesentliches Merkmal ist dabei ist das eine mögliche Lösung auch erreichbar ist wenn bestimmte Grundkonzepte nicht oder nur teilweise Verstanden wurden. Diese Anpassungsfähigkeit trägt dazu bei, eine angenehme Lernatmosphäre und Erfolgserlebnisse zu schaffen, zudem werden mit dem Schüler-PDF allgemeine Hinweise zum verbessern des Lernprozesses vermittelt welche dem Endbenutzer in Zukunft helfen werden.

5.1.2.5 Weitere Werkzeuge

Teil des Lernprozesses sind verschiedene Hilfswerkzeuge aus der Programmierung welche zur Bearbeitung der Übungen benötigt werden. Dazu gehört ein Texteditor der dazu genutzt wird den C++ Code zu bearbeiten, sowie ein Terminal mit dem man einen beliebigen C++ Compiler bedient um den angefertigten Code in eine von der Swing-Anwendung ausführbare Datei zu übersetzen.

5.1.3 Benutzergruppen

Die Hauptbenutzergruppen der Anwendung sind Schüler im Alter von **13 bis 18 Jahren**, die Grundlagen der Programmierung erlernen möchten. Die Benutzeroberfläche und die Lerninhalte sind so gestaltet, dass sie für Anfänger ohne Vorkenntnisse leicht verständlich sind.

5.1.4 Anwendungsumgebung

Die Anwendung ist für moderne Betriebssysteme wie **Windows, Linux** und **MacOS** optimiert. Sie erfordert **Java Version 17+** bzw. **Java JDK 21** sowie einen **C++ Compiler**, in der Schüler-PDF findet **clang** Verwendung aber es ist auch jeder andere Compiler zulässig wie **gcc** oder **g++**. Für manche Features der Schüler-PDF ist eine stabile Internetverbindung nötig. Die Anwendung wurde für hochauflösenden Bildschirmen optimiert, bei niedrig auflösenden Bildschirmen könnte es zu Darstellungsproblemen kommen, weshalb eine Auflösung von wenigstens 720p empfohlen wird.

5.1.5 Benutzerdokumentation

Dem Benutzer wird eine essentielle Schüler-PDF bereitgestellt, die die Grundlagen der Programmierung und Farbdarstellung erklärt sowie allgemeine Hinweise zum verbessern des Lernprozesses vermittelt.

5.1.6 Annahmen und Abhängigkeiten

Bei der Entwicklung wurde angenommen, dass das Endgerät der Benutzer bereits über eine aktuelle Java Distribution verfügen, sowie eine Entwicklungsumgebung für C++ eingerichtet ist, was die Hauptsystemanforderung darstellt. Wenn nicht dann finden sich in der Lehrer-PDF weiterführende Links um diese Voraussetzungen sicherzustellen. Darüber hinaus wird davon ausgegangen, dass Benutzer über grundlegende Computer- und Browserkenntnisse verfügen und mit den bereitgestellten Tools und Funktionen umzugehen zu wissen, aber möglicherweise keine Erfahrung mit Programmierung haben.

Die Anwendung hat keine weitere Abhängigkeiten wie externe Bibliotheken oder Module, da die Swing Anwendung fertig kompiliert ausgeliefert wird.

5.2 Anforderungen

Im Laufe des Projektes wurden den Rahmenbedingungen entsprechende Anforderungen festgelegt.

5.2.1 Rahmenbedingungen

Ausarbeiten eines Programmes und Arbeitspakets für Veranstaltungen für Schüler und Erstsemester

- Funktioniert auf allen gängigen Betriebssystemen (**Soll-Ziel**)
- Die Bearbeitung des Arbeitspaketes soll ungefähr 90 Minuten beanspruchen

5.2.2 Funktionale Anforderungen

Das Programm soll .out Dateien einlesen und mit einer Java Swing Anwendung grafisch ausgeben

- Die Projektarbeit von Nurullah Damla soll angepasst und überarbeitet werden
- Kann angepasst werden um in künftigen C/C++ Praktiken unterstützend in Aufgaben verwendet zu werden (**Kann-Ziel**)
- Ausgefüllte Formen sollen schneller dargestellt werden
- Soll nicht im Developer View starten
- Dreiecke und Kreise sollen ausfüllbar sein
- Linienbreite von Rechtecken soll Variable sein
- Layering soll nach Reihenfolge passieren
- Code in der Anwendung schreibbar (**Kann-Ziel**)
- Anwendung als executable (**Kann-Ziel**)
- Bessere Übersicht bei geringer Skalierung (**Kann-Ziel**)

Es soll ein Dokument für Schüler ausgearbeitet werden

- Soll den Schülern einen Einblick ins Programmieren geben und Interesse daran wecken
- Es soll einen 'Aha' Effekt geben
- Es soll das Gefühl vermittelt werden das 'Fürs Leben gelernt' wurde

Es soll ein Dokument für Lehrende ausgearbeitet werden

- Es muss mindestens eine Musterlösung geben
- Soll Installationsanweisungen enthalten
- Soll Lehrendem helfen den Schülern zu helfen

5.3 Umsetzung

5.3.1 Triangle/Circle/Rectangle Fill

Bei der ersten Besichtigung des Programmes ergab sich das einige Funktionen nicht gegeben waren oder nicht den Anforderungen entsprachen. Darunter fiel, dass die Funktion der 'floodfill()' -Funktion nur bei Rectangle-Objekten funktionierte und, dass das tatsächliche Füllen zu lange dauerte. Dies wurde in der DrawGUI.cpp behoben in dem die Funktion durch 'fill()' ersetzt wurde die für alle Objekte anwendbar. Im Java Code wurden dafür Standard-Methoden der java.awt.Graphics2D Bibliothek verwendet. Diese unterscheiden sich in der .out durch einen davor gesetzten Buchstaben d für draw und f für fill.

5.3.2 Zeichnen

Es gibt zwei C++ Dateien. Einmal DrawGUI.cpp mit Grundfunktionen und eine Shapes_Main.cpp mit #include "DrawGUI.cpp" die auf diese Grundfunktionen weiter aufbaut und die Main enthält.

Um etwas darzustellen kann man die folgende Objekte und Funktionen verwenden:

```
setPixel(2, 2, 50, 50, 50);

StringText* stringText = new StringText(25, 10, "Moin", "Comic Sans", 12, 1, 50, 50, 255);
stringText->draw();

Line* line = new Line(45, 55, 25, 15, 100, 0, 255, 5);
line->draw();

Triangle* triangle = new Triangle(11, 11, 22, 22, 11, 22,
    100, 100, 10, 5);
triangle->draw();
triangle->fill();

Circle* circle = new Circle(0, 0, 200, 0, 0, 0, 5);
circle->draw();
circle->fill();

Rectangle* rectangle = new Rectangle(0, 20, 80, 30, 255,
    255, 255, 5);
rectangle->draw();
rectangle->fill();
```

5.3.3 Developer View

5.3.3.1 Developer View

Der Developer View ist nun nicht mehr bei Start der Anwendung aktiv. Dies lässt sich in farbzauber/gui/FarbzauberFrame.java wieder rückgängig machen.

```
public boolean developerView = false;
```

Des Weiteren wurde durch auskommentieren die option genommen den Developer View über die Benutzeroberfläche der Anwendung zu aktivieren. Dies lässt sich in farbzauber/menueBar/MenuBarForGUI.java wieder rückgängig machen.

```
//this.file.addSeparator();
//this.file.add(developerView);
```

5.3.4 C++ zu Java

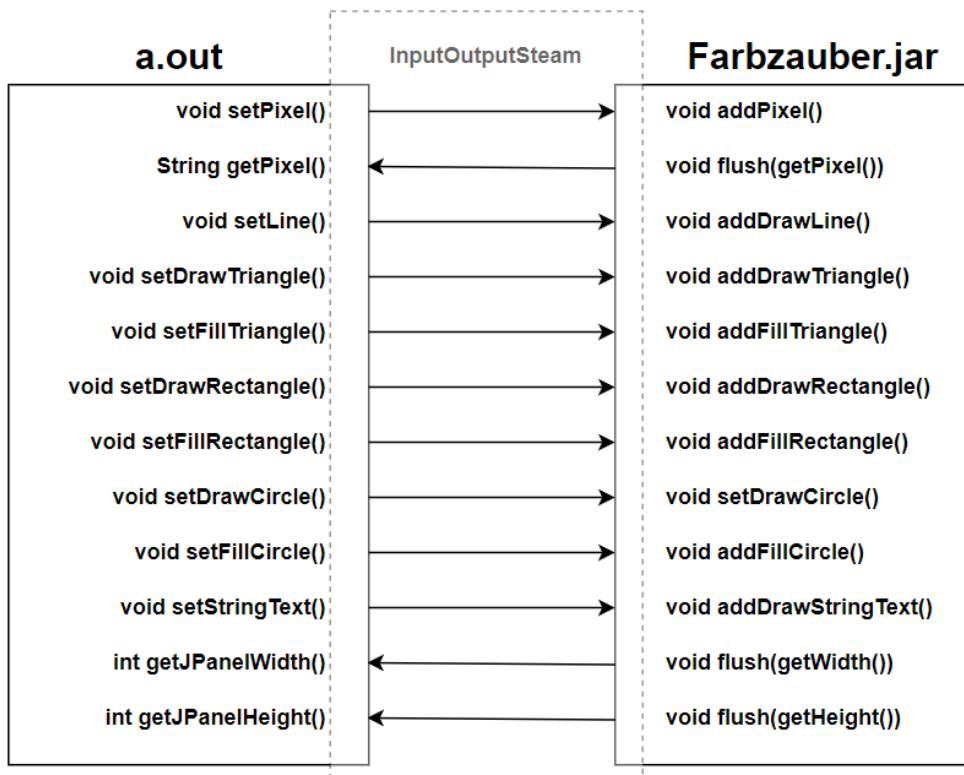


Abbildung 13: inputOutputStream

5.4 Evaluation

Bis zum Ende des Projektes wurden die Anforderungen verschieden gelöst. Hier geht es um einen Vergleich mit den Anforderungen und was in Zukunft noch daraus gemacht werden kann.

5.4.1 Vergleich zu den Anforderungen

Ausarbeiten eines Programmes und Arbeitspakets für Veranstaltungen für Schüler und Erstsemester

Die Anwendung läuft auf Windows, Linux und Mac und damit auf den drei gängigsten Betriebssystemen. Die Möglichkeit verschiedene Formen anpassbarer Position, Form und Farbe mit den erklärten Programmierbeispielen nutzen zu können bietet nach dem Versuch ein Bilderbeispiel darzustellen genügend Möglichkeiten mit eigenem Experimentieren beliebig viel Zeit damit zu verbringen.

Das Programm soll .out Dateien einlesen und mit einer Java Swing Anwendung grafisch ausgeben

Es die bereitgestellte Projektarbeit von Nurullah Damla wurde wie in den folgenden Punkten erwähnt angepasst und Überarbeitet, da es sich nicht um eigenen Code handelte musste man sich des öfteren wieder einlesen. Ob die Anwendung tatsächlich ausreichend angepasst wurde um auch in künftigen C/C++ Praktika Verwendung zu finden ist noch ungewiss. Die Anwendung starten nicht mehr im Developer View was auch eine bessere Übersicht bei geringer Skalierung zur Folge hat. Dreiecke und Kreise lassen sich jetzt auch ausgefüllt darstellen und Rechtecke haben nun eine Variable Linienbreite. Das Layering wurde überarbeitet wodurch nun die zuletzt dazustellende Form umverdeckt ist. Die Anwendung als Executable zu kennzeichnen wurde verworfen da es ausreichend war eine ausführbare JAR bereitzustellen. Das Ziel hierbei war es den Schüler die bekanntere .exe Endung zu geben. Neben dem fehlenden nutzen durch die erwähnung der JAR im Schülerdokument kommen auch Kompatibilitätsprobleme durch verschiedene Betriebssysteme dazu. Aus Zeitmangel wurde es nicht umgesetzt den Code innerhalb der Anwendung schreiben zu können. Des Weiteren wurde aus den dazu getätigten Recherchen nicht klar ob dieses Ziel ohne zu großem Mehraufwand verwirklicht werden kann.

Es soll ein Dokument für Schüler ausgearbeitet werden

Im Schülerdokument werden erste Programmierbegriffe erklärt und Beispiele dazu gegeben. Zusammen mit dem anschließenden Ausprobieren sollte ein erster Einblick bestehen und gegebenenfalls auch das Interesse am Programmieren geweckt werden. Durch die Erklärung von RGB bzw. der additiven Farbmischung sollte ein 'Aha' Effekt entstehen der auch im Alltag auftreten kann. Ein

Einblick in die Programmierung bei etwas scheinbar simplem wie etwa Formen bzw. Bilder darstellen zu lassen könnte das Gefühl vermitteln 'Fürs Leben gelernt' zu haben was bzw. wieviel doch bei Mediengeräten wie PC dahinter steckt.

Es soll ein Dokument für Lehrende ausgearbeitet werden

Es werden zwei Musterlösungen bereitgestellt, eines für ein Weihnachtsbild und eines für ein Mandala.

6 Programmieren lernen mit dem MIT App Inventor

In diesem Abschnitt wird auf das Szenarienpaket MIT App Inventor eingegangen. Dabei handelt es sich um die grundlegende Vermittlung von Programmierelementen, mittels in MIT App Inventor erstellten Aufgaben. Es gibt dabei eine Hauptaufgabe und mehrere, kleine Einführungsaufgaben die zum erfolgreichen Umsetzen und Verstehen der Hauptaufgabe führen sollen.

In den Folgenden Unterkapiteln wird auf die Anforderungen, die Umsetzung und Schwierigkeiten sowie das Resultat eingegangen.

Wichtig:

Diese Aufgabe wurde inspiriert von: <https://mit-cml.github.io/yrtoolkit/yr/tutorials/mytodolist.html>
unter der Creative Commons 4.0 Lizenz (<https://creativecommons.org/licenses/by-sa/4.0/>)

6.1 Produktbeschreibung

In diesem Abschnitt wird darauf eingegangen aus was sich das Szenario zusammensetzt.

6.1.1 Der MIT App Inventor

Bei dem MIT App Inventor (folgend MITAI genannt) handelt es sich um eine von dem Massachusetts Institute of Technology (MIT) erstellten Anwendung um jungen Menschen das Erlernen und die Denkweise vom Programmieren näher zu bringen. Mit Hilfe des MITAI können Android Anwendungen für einen Android Emulator oder ein Android Gerät (mittels einer zugehörigen Companion App) erstellt werden.

6.1.1.1 Funktionen

Mit dieser Anwendung kann eine Komplette Android Anwendung auf einer graphischen Oberfläche erstellt werden, welche einfach zu verstehen ist. Der MITAI unterteilt sich dabei in zwei große Bereiche:

Bereich 1: Designer In diesem Bereich kann mittels *Drag and Drop* das Aussehen der zu erstellenden Anwendung erstellt und bearbeitet werden. Darunter zählen z.B. das Einfügen von Schaltflächen, Textfeldern, etc., das Anordnen und Anpassen dieser als auch das Hinzufügen von nicht sichtbaren Elementen wie z.B. einer Datenbank.

Bereich 2: Blöcke Dieser Bereich ist für die Erstellung der Logik bzw. für das Programmieren da. Das Programmieren erfolgt mittels vorgefertigten Programmblöcken und ist somit graphisch mittels *Drag and Drop* umsetzbar. Dadurch erspart man sich Tippfehler und kann direkt aus einer Liste verfügbarer Programmstücke auswählen. Zusätzlich ist die Hierarchische Blockansicht besser zum Verständnis des geschriebenen Programms für Anfänger.

6.1.1.2 Bedienung

Der MITAI wird über eine Web-Oberfläche gesteuert. Die meisten Funktionen sind mittels *Drag and Drop*, *Drop Down* oder *Check Boxen* realisiert. Somit ist die Steuerung der einzelnen Elemente der Anwendung sehr Anfänger freundlich gehalten.

6.1.2 Aufgaben

Die Aufgaben die bearbeitet werden sollen sind auf einem PDF Dokument festgehalten. Dort findet man eine Schritt für Schritt Anleitung, die einen durch die verschiedenen und aufeinander aufbauenden Aufgaben leitet. Die ersten Aufgaben beziehen sich dabei jeweils auf einzelne Elemente der Programmierung die zum Schluss dann in die Anwendung To-Do Liste münden. Die To-Do Liste ist eine einfache Anwendung in welcher man Einträge in Textform (String) in einer Liste hinzufügen kann. Die Anwendung nutzt lediglich Listen, If-Bedingungen, TinyDB, Strings und Integer (Index) und ist somit klein gehalten.

6.2 Anforderungen

In diesem Abschnitt geht es um die geplanten Anforderungen welche sich im laufe des Projektes herauskristallisiert haben.

6.2.1 Rahmenbedingungen

- Mehrere Personen (z.B Schüler) befinden sich in einem Raum mit jeweils einem PC
- Zusätzlich befindet sich mindestens eine betreuende Person in der Gruppe
- Zeitlicher Rahmen von 90 Minuten

6.2.2 Aufgabe

- Einfach zu verstehen
- Im Alltag nutzbar
- Mehrere Programmierkonzepte werden näher gebracht

6.2.3 Dokument für Schüler

- Soll die Aufgaben für Anfänger einfach verständlich erklären
- Bringt Struktur in die Veranstaltung
- Kann als Nachschlagewerk benutzt werden

6.2.4 Dokument für Lehrende

- Es muss mindestens eine Musterlösung geben
- Soll Installationsanweisungen enthalten
- Soll Betreuende helfen den Schülern zu helfen

6.3 Umsetzung

In diesem Abschnitt wird darauf eingegangen wie mit Hilfe des MITAI die Aufgabe für die Schüler konstruiert wurde.

6.3.1 Schwierigkeiten

Die größte Schwierigkeit bestand darin sich auf eine Technologie zu einigen. Nach dem die Wahl auf den MITAI fiel, musste man sich überlegen welche Anwendung man zum Schluss für die Schüler erstellen lassen möchte. Es wurden im Laufe des Projektes wiederholt Vorschläge gemacht die es zu bewerten galt. Dadurch kam es zu Verzögerungen beim Bearbeiten der Aufgabe.

6.3.2 Art der zu Programmierenden Anwendung

Zuerst musste festgelegt werden welche Art Anwendung die Personengruppe erstellen soll, um den Anforderungen so gut es geht gerecht zu werden. Erste Idee war ein Vokabel Trainer welche aus Komplexitäts Gründen später verworfen wurde und gegen eine To-Do Liste ausgetauscht wurde, um den später erstellten Block-Code einfacher zu halten. Ein weiterer Ansatz war die Erstellung einer Anwendung welche dem Spiel *Simon* (zu Deutsch Senso) nachempfunden sein sollte. Dies war allerdings nicht ohne weiteres umsetzbar daher der MITAI keine eingebaute sleep-Funktion hat, eine umständlichere Umsetzung eines sleeps wäre möglich gewesen, jedoch hätte es zu Verwirrung des Nutzers führen können und wir wollten die Nutzer nicht von der eigentlichen Aufgabe ablenken.

6.3.3 Aufbau der To-Do Aufgabe

Die Aufgabe wurde so erstellt, dass man den MITAI öffnet und dort über mehrere kleine Anwendungen auf einzelnen *Bildschirmen* die Konzepte Bedingungen, Listen und Speicher (Datenbankeinbindung) kennen lernen kann. Diese werden anschließend dazu benötigt die To-Do Listen Aufgabe zu bearbeiten. Durch diese schrittweise Einführung soll das Lernen erleichtert werden. Zusätzlich lernt man dadurch große Aufgaben in kleinere Aufgaben zu zerlegen und so das Arbeiten an diesen zu vereinfachen.

6.4 Resultat

Die erstellten Aufgaben erfüllen die Voraussetzungen. Sie sind einfach zu verstehen, im Alltag anwendbar und Zeitlich schaffbar.

6.4.1 Zukunft

Auf der Basis des lokalen MITAI kann die Aufgabe als Inspiration genutzt werden um weitere Aufgaben ähnlicher Art zu erstellen.

7 Fazit

Zum Abschluss dieser Dokumentation wird das Projekt noch einmal kurz zusammengefasst und die wichtigsten Erkenntnisse genannt.

7.1 Zusammenfassung

Die Szenarienpakete gewannen erst gegen Mitte des Projekts an Momentum, nachdem sie anfänglich nur langsam vorankamen. Trotzdem haben alle Szenarienpakete ein abgeschlossenes Stadium erreicht, welches auslieferbar ist. Es ist jedoch zu beachten, dass nicht alle von uns gewünschten Kriterien vollständig erfüllt wurden.

7.2 Gewonnene Erkenntnisse

Während des Projekts haben wir wichtige Erfahrungen gesammelt. Regelmäßige Treffen und eine effektive Kommunikation erwiesen sich als entscheidend für einen reibungslosen Projektverlauf. Die Nutzung von GitLab erleichterte die Zusammenarbeit und das Tracking von Änderungen. Ebenso wichtig war die regelmäßige Abstimmung mit dem Auftraggeber, um potenzielle Missverständnisse in der Softwareentwicklung zu vermeiden. Die Kommunikation mit dem Kunden hätte durch die Verwendung gemeinsam abgestimmter Protokolle der Treffen verbessert und klarer gestaltet werden können. Es gestaltete sich schwierig, gruppenübergreifend zusammenzuarbeiten oder sich gegenseitig zu unterstützen, da die verwendeten Programmierumgebungen und Programmiersprachen erheblich voneinander abwichen.

Literatur

- [1] Brython. Brython. URL <https://brython.info/index.html>.
- [2] M. Haverbeke. Extensible code editor. URL <https://codemirror.net>.
- [3] M. Bostock and Observable Inc. d3. URL <https://d3js.org/>.
- [4] Python Software Foundation. Python logo, . URL <https://www.python.org/community/logos/>.
- [5] Python Software Foundation. Turtle graphics, . URL <https://docs.python.org/3/library/turtle.html>.
- [6] S. Sangmin. qrcode.js. URL <https://davidshimjs.github.io/qrcodejs/>.
- [7] WOJTEK SAS. gofile.io. URL <https://gofile.io/welcome>.

Abbildungsverzeichnis

1	Darstellung der Webanwendung	3
2	Struktur der Webanwendung	10
3	Darstellung der Webanwendung	24
4	Darstellung der gekürzten Webanwendung	25
5	Play_button	29
6	Options_button	29
7	Exit_button	29
8	Delete_button	30
9	Start_button	30
10	Help_button	30
11	Back_button	31
12	Darstellung der Swing-Anwendung	34
13	inputOutputStream	40
14	Python-Code umwandeln in Computersprache [4]	xvii
15	Lösung Level 1	lii
16	Lösung Level 2	lii
17	Lösung Level 3	liii

Listings

1	load_level()	13
2	read_level_from_json_file()	13
3	run_code()	14
4	redirect_prints_and_errors_to_console()	15
5	write_console()	15
6	extract_relevant_part_of_error_message()	16
7	Hilfsfunktion vom Script brython_turlte_init.sh	vii
8	level_n JSON-Datei	viii
9	Hilfsfunktion vom Script html_2_string.sh	viii
10	Hilfsfunktion vom Script python_2_string.sh	ix
11	Level 0	x
12	Level 1	xi
13	Level 2	xi
14	Level 3	xii
15	Level 4	xii
16	Level 5	xiii
17	Level 6	xiii
18	Level 7	xiv
19	Lösung: Aufgabe 0	xxxiv
20	Lösung: Aufgabe 1	xxxv
21	Lösung: Aufgabe 0	xxxvi
22	Lösung: Aufgabe 0	xxxvi
23	Lösung: Aufgabe 0	xxxvii
24	Lösung: Aufgabe 0	xxxviii
25	Variablen in Python	xl
26	Rechnen mit Variablen	xl
27	Ausgeben von Variablen	xl
28	Aufbau der for Schleife	xli
29	Konsolenausgabe	xli
30	Gemischte Arrays	xlii
31	Arbeiten mit Arrays	xlii
32	Werte eines Arrays ausgeben	xlii
33	a größer 20	xliii
34	a größer/kleiner 20	xliii
35	a größer/kleiner/gleich 20	xliii
36	Aufbau einer Funktion	xliv
37	Funktionsbeispiel	xliv
38	Einzelnen Ton abspielen	xlv
39	Mehrere Töne gleichzeitig abspielen	xlv
40	Töne nacheinander abspielen lassen	xlv
41	ABC Notation	xlvi
42	ABC Notation als String	xlvi
43	ABC Notation mit play() abspielen lassen	xlvi

Index

A Anhang - Learn Python with Turtle Graphics

A.1 Ablaufdiagramme zu Schnittstellen

Level laden (Schnittstelle)

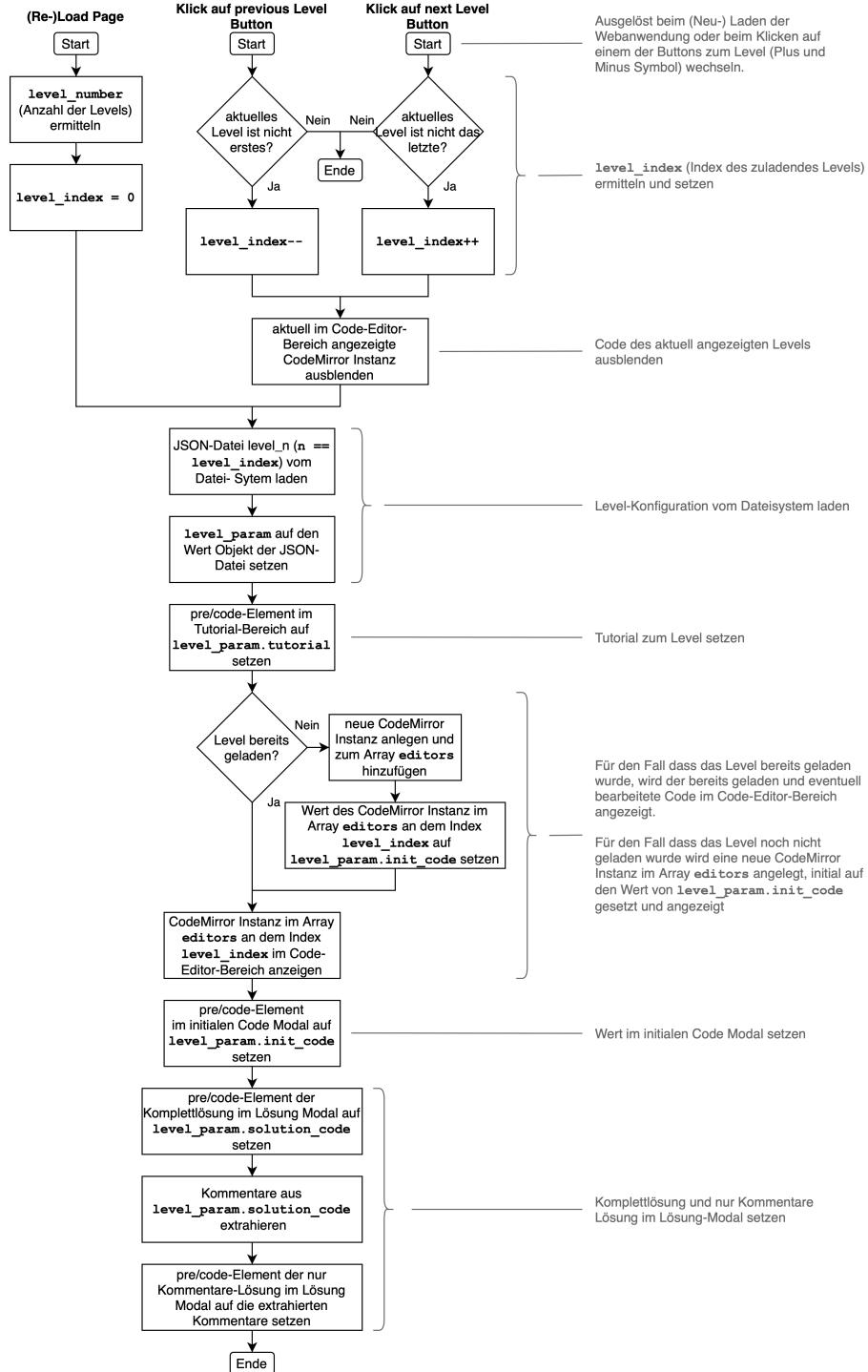


Abbildung A.1.1: Ablaufdiagramm Level laden

Code ausführen (Schnittstelle)

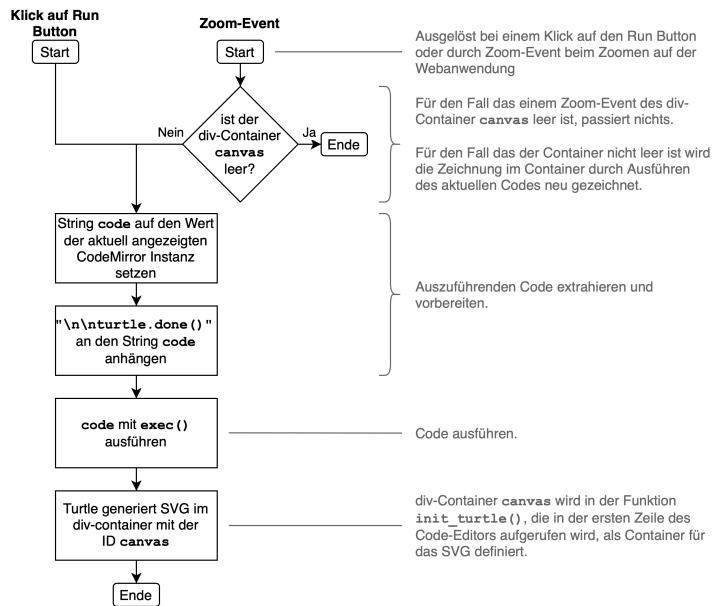


Abbildung A.1.2: Ablaufdiagramm Code ausführen

Konsolenausgabe (Schnittstelle)

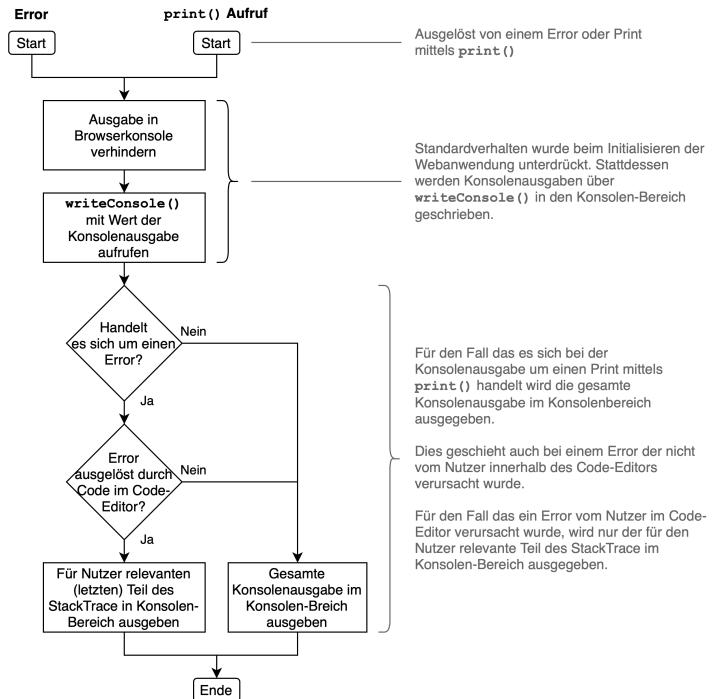


Abbildung A.1.3: Ablaufdiagramm Konsolenausgabe

A.2 Lehrer-PDF

In dieser Lehrer-PDF zum Szenario und Webanwendung "Learn Python with Turtle Graphics" wird die Installation und Inbetriebnahme der Webanwendung beschrieben. Außerdem wird eine Anleitung zum Generieren neuer Levels gegeben. Des Weiteren enthält diese das Passwort zum Anzeigen von Musterlösungen und wie dieses verändert werden kann. Außerdem werden Musterlösungen zu allen bestehenden Leveln gegeben wie auch typische Fehlermeldungen und deren Behebung erläutert.

Inhaltsverzeichnis:

1. Installation und Inbetriebnahme
2. Erstellung von neuen Levels
 - 2.1 Level Struktur
 - 2.2 Tutorial erstellen
 - 2.3 Code-Hinweise für Tutorial, initialen Code und Musterlösung erstellen
3. Passwort für Musterlösung
4. Musterlösungen
5. Typische Fehlermeldungen und Quick-Help

Installation und Inbetriebnahme

Um eine reibungslose Installation und Inbetriebnahme der Webanwendung zu gewährleisten, wird das Bash-Script `brython_turtle_init.sh` zur Verfügung gestellt. In [Listing 7](#) ist die Ausgabe der Hilfsfunktion mit Anweisungen zum Einsatz des Scripts dargestellt.

```
Listing 7: Hilfsfunktion vom Script brython_turtle_init.sh
Usage: ./brython_turtle_init.sh [options]
Initializes the Learn-Python-with-Turtle-Graphics repository
      ,
starts the Python HTTP-server and opens webapp.

Options:
  -dir, --directory DIRECTORY Specify the target directory.
    Default is 'Learn-Python-with-Turtle-Graphics'.
  -h, --help Display this help message.
```

Dieses cloned das GitHub-Repository in einen angegebenen directory oder ein default-directory, startet den Python HTTP-Server und öffnet die Webanwendung im Standard-Browser. Für den Fall das bereits gecloned wurde und dieses lokal nicht auf dem neusten Stand ist wird gepulled.

Erstellung von neuen Levels

Das Hinzufügen neuer Level sowie das Bearbeiten bestehender Level ist unkompliziert. Im folgenden wird die Struktur eines Levels wie auch Skripte zum Erstellen dieser beschrieben.

Level Struktur

Zum Hinzufügen eines neuen Levels muss dieses lediglich als JSON-Datei im Ordner "levels" abgelegt werden. Zum Bearbeiten eines Levels wird die jeweilige JSON-Datei bearbeitet. Die Bezeichnung eines Levels sollte "level_n" lauten, wobei "n" für die Nummer des jeweiligen Levels steht. Die Struktur einer solchen JSON-Datei ist in [Listing 8](#) dargestellt.

```
Listing 8: level_n JSON-Datei
{
    "tutorial": "",
    "init_code": "t = init_turtle(turtle) # Diese Zeile bitte
                 nicht entfernen oder ändern\n\n",
    "solution_code": "t = init_turtle(turtle) # Diese Zeile
                     bitte nicht entfernen oder ändern\n\n"
}
```

Ein Level besteht aus einem Tutorial, einem initialen Code und einer Musterlösung. Diese sind als Strings angeben. Der String `tutorial` enthält HTML-Code. `init_code` und `solution_code` enthalten Python-Code. Diese müssen zunächst in das Format eines einzeiligen Strings gebracht werden. Das Konvertieren sowie das Maskieren von Sonderzeichen geschieht in zwei Bash-Scripts, die im Folgenden erläutert werden.

Tutorial erstellen

Der String `tutorial` ist ein beliebiger HTML-Code, der mithilfe des Bash-Scripts `html_2_string.sh` generiert wird. In [Listing 9](#) ist die Ausgabe der Hilfsfunktion mit Anweisungen zum Einsatz des Scripts dargestellt.

```
Listing 9: Hilfsfunktion vom Script html_2_string.sh
Usage: ./html_2_string.sh [options] HTML_FILE
Converts an HTML file to a single-line string.

Options:
    -out, --output_file FILENAME Specify the output file.
    -help, --help Display this help message.

Example:
    ./html_2_string.sh input.html
    ./html_2_string.sh input.html -out output.txt
```

Die übergebene HTML-Datei kann beliebige HTML-Elemente enthalten. Dennoch existieren für ein konsistentes Level Design einige empfohlene Styleguidelines. Diese sind in [Tabelle 1](#) dargestellt und nutzen vordefinierte CSS-Klassen. Für zusätzliches Styling können direkt in HTML-Elementen weitere Stilregeln ohne Klassendefinitionen angegeben werden.

Aufgabenstellungen und Text-Hinweise werden als String ohne Anführungsstriche, anstelle der Platzhalter, angegeben. Nach dem Konvertieren der HTML-Datei in den generierten String, werden Code-Hinweise anstelle des Platzhalters eingesetzt. Zunächst müssen auch die Code-Hinweise in einen String gewandelt werden. Dies erfordert ein weiteres Script, das im folgenden Abschnitt erläutert wird.

Tabelle 1: Tutorial Styleguidelines

Typ	HTML Code
Aufgabenstellung	<p class="p-font">AUFGABENSTELLUNG</p>
Text-Hinweis	<p class="p-font">TEXT-HINWEIS</p>
Code-Hinweis	<pre class="code-hint"><code>CODE-HINWEIS</code></pre>

Code-Hinweise für Tutorial, initialen Code und Musterlösung erstellen

Das Bash-Script `python_2_string.sh` wird verwendet, um Python Dateien in einen String zu konvertieren. Dieses wird auf Code-Hinweise sowie auf `init_code` und `solution_code` für die JSON-Datei angewandt. In [Listing 10](#) ist die Ausgabe der Hilfsfunktion mit Anweisungen zum Einsatz des Scripts dargestellt.

```
Listing 10: Hilfsfunktion vom Script python_2_string.sh
Usage: ./python_2_string.sh [options] PYTHON_FILE
Converts a Python file to a single-line string.

Options:
  -out, --output_file FILENAME Specify the output file.
  -h, --help Display this help message.

Example:
  ./python_2_string.sh input.py
  ./python_2_string.sh input.py -out output.txt
```

Die als Input übergebene Python Datei kann hierbei beliebigen Code enthalten. Beim Importieren von Python Modulen mittels `import <module>` sollte

überprüft werden, ob diese möglicherweise von Brython nicht unterstützt werden. Die erste Zeile der Strings `init_code` und `solution_code` sollte folgendermaßen lauten:

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern
```

Diese Zeile initialisiert Turtle Graphics. Ohne diese ist Turtle Code nicht ausführbar und wirft folgenden Error:

```
line 7, in Code-Editor NameError: name 't' is not defined. Did you mean '[object Object]'?
```

Passwort für Musterlösung

Lösungen können in zwei Schritten angezeigt werden. Schüler haben immer Zugriff auf die Kommentare (ohne Code) einer Musterlösung. Für die Musterlösung mit Code brauchen Schüler ein Passwort. Diese Passwort lautet aktuell "Passwort". Dieses lässt sich konfigurieren indem der String `password` in Zeile 6 der Datei `/modules/solution.py` angepasst wird.

Musterlösungen

Im diesem Abschnitt werden zu jedem Level Musterlösungen mit Code und Kommentaren dargestellt.

Level 0

```
Listing 11: Level 0
1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern
2
3 t.color("red") # Zeichenfarbe wird auf Rot gesetzt
4 t.forward(100) # Bewegt die Turtle 100 Einheiten nach vorne
5 t.right(90) # Dreht die Turtle um 90 Grad nach Rechts
6 t.color("blue") # Zeichenfarbe wird auf Blau gesetzt
7 t.backward(100) # Bewegt die Turtle 100 Einheiten nach hinten
8 t.left(45) # Dreht die Turtle um 45 Grad nach Links
```

Level 1

Listing 12: Level 1

```

1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
   oder ändern
2
3 length = 100      # Kantenlänge
4 angle = 90        # Winkel
5 color = 'red'     # Farbe
6
7 t.color(color)    # ändert Farbe des Stiftes und der
   Schildkröte auf die Farbe in der Variable color
8
9 t.begin_fill()
10
11 t.forward(length) # 100 Schritte nmachvorne
12 t.right(angle)    # Schildkröte um 90 Grad nach rechts
   drehen
13
14 # das gleiche wie oben noch 3 mal
15 t.forward(length)
16 t.right(angle)
17
18 t.forward(length)
19 t.right(angle)
20
21 t.forward(length)
22 t.right(angle)
23
24 t.end_fill()
```

Level 2

Listing 13: Level 2

```

1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
   oder ändern
2
3 length = 100      # Kantenlänge
4 angle = 90        # Winkel
5 text_style=('Arial', 20, 'normal') # Variable mit Text-
   Style
6
7 for i in range(4): # darunter eingerückter Code wird
   viermal wiederholt
8   t.color('red')    # ändert Farbe des Stiftes und der
   Schildkröte auf rot
9   t.write(i, font=text_style) # schreibt die Zahl "i" in die
   Ecke mit dem Style aus einer Variable
10  t.color('blue')   # ändert Farbe des Stiftes und der
   Schildkröte auf blau
11
```

```

12 t.forward(length) # 100 Schritte nach vorne
13 t.right(angle)    # Schildkröte um 90 Grad nach rechts
14      drehen
15 t.hideturtle()    # versteckt die Schildkröte

```

Level 3

Listing 14: Level 3

```

1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
2      oder ändern
3
4 length = 100      # Kantenlänge
5 corner_num = 6     # Anzahl der Ecken
6 angle = 360 / corner_num # Winkel berechnet mit Anzahl der
7      Ecken
8
9 for i in range(corner_num): # for-Schleifenkopf, darunter
10      eingerückter Code wird so oft wiederholt wie in der
11      Variable mit Anzahl der Ecken angegeben
12      if i % 2 == 0:      # Wenn Zahl "i" durch 2 teilbar ist ...
13          t.color('red')   # ist die Schildkröte rot
14      else:             # Sonst ...
15          t.color('blue') # ist die Schildkröte blau
16
17 t.forward(length) # Kantenlänge nach vorne
18 t.right(angle)    # Schildkröte um den Wert Winkel
19      Variable nach rechts drehen

```

Level 4

Listing 15: Level 4

```

1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
2      oder ändern
3
4 color_array=['red','green','blue','turquoise','yellow','#3
5      c79b8'] # Farb-Array mit verschiedenen Farben
6 for i in range(6):      # for-Schleife mit einer 6-fachen
7      Wiederholung erstellen. Beginnned mit 0
8      if(i%2==1):        # Wenn die Zahl ungerade ist, dann
9          wird die Turtle versteckt
10         t.hideturtle()
11      else:             # Wenn die Zahl gerade ist, dann wird die
12          Turtle wieder gezeigt
13         t.showturtle()
14      t.speed(i+1)       # Turtle-Speed abhängig des Index erhö
15         hen

```

```

10 t.color(color_array[i])    # Setzt die Farbe abhängig des
   Index durch Auswahl im Array
11 t.forward(50)           # Bewegt die Turtle 50 Einheiten nach
   vorne
12 t.right(60)            # Dreht die Turtle um 60 Grad nach
   Rechts

```

Level 5

```

Listing 16: Level 5
1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
   oder ändern
2
3 import random
4
5 def draw_form(length=100, angle=90, lines=4): # Funktion
   definiert mit 3 vorgelegten Variablen.
6   color_array=['red','green','blue','turquoise','yellow','#3
   c79b8'] # Farb-Array mit verschiedenen Farben
7   for x in range(lines):
8     t.speed(x+1) # Turtle-Speed
9     if(random.randint(1,2)==1): # Turtle wird zufällig
   Sichtbar oder nicht mit einer 50/50-Chance
10    t.hideturtle() # Turtle wird versteckt
11  else:
12    t.showturtle() # Turtle wird Sichtbar
13    color = color_array[random.randint(0,5)]
14    t.color(color) # Setzt die Farbe der Turtle zufällig
   aus dem Array
15
16    t.forward(length) # Bewegt die Turtle um length
   Einheiten nach vorne
17    t.right(angle) # Dreht die Turtle um angle Grad nach
   Rechts
18  return color
19
20 print(draw_form(angle=60, lines=6))

```

Level 6

```

Listing 17: Level 6
1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
   oder ändern
2
3 def form(corner_num=30, length=10, color='blue'): #
   Funktionskopf, darunter eingerückter Code wird beim
   Aufruf der Funktion ausgeführt

```

```

4   angle = 360 / corner_num # Winkel berechnet mit Anzahl
5     der Ecken
6   t.color(color)          # färbt Schildkröte mit übergebener
7     Farbe
8   for _ in range(corner_num): # for-Schleifenkopf, darunter
9     eingerückter Code wird so oft wiederholt wie in der
10    Variable mit Anzahl der Ecken angegeben
11    t.forward(length)       # Kantenlänge nach vorne
12    t.left(angle)          # Schildkröte um den Wert Winkel
13    Variable nach links drehen
14
15  for x in range(3): # for-Schleifenkopf, darunter eingerü-
16    ckter Code wird 3 mal wiederholt
17    for y in range(3): # for-Schleifenkopf, darunter eingerü-
18    ckter Code wird 3 mal wiederholt
19    t.goto(100 * x, 100 * y) # zu Koordinaten gehen
20    form()                 # Aufruf der Funktion zum Form
21    zeichnen
22
23
24
25
26

```

Level 7

Listing 18: Level 7

```

1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
2   oder ändern
3
4
5 length = 3
6 last = 0
7
8 def right():
9   t.color("red")
10  for _ in range(10):
11    t.forward(length)
12    t.right(9)
13
14 def left():
15  t.color("yellow")
16  for _ in range(10):
17    t.forward(length)
18    t.left(9)
19
20 def forward():
21  t.color("green")
22  t.forward(20)
23
24 def teleport(x=0,y=0):
25  t.penup()
26  t.goto(x,y)

```

```
27     t.pendown()
28
29 for _ in range(40):
30     x = random.randint(0, 6)
31     if x == 0:
32         right()
33     elif x == 1 or x == 2:
34         left()
35     elif x == 3 or x == 4 or x == 5:
36         forward()
37     elif x == 6:
38         teleport(0,0)
```

Level 8

Keine Lösung, da es sich um eine Sand-Box handelt.

Typische Fehlermeldungen und Quick-Help

Im Folgenden sind typische Fehlermeldung wie auch Hilfe beim Auftritt dieser dargestellt.

1. Error:

```
line 7, in Code-Editor NameError: name 't' is not defined.
Did you mean '[object Object]'?
```

Quick-Help:

Zeile 1 wurde entfernt oder verändert. Zeile 1 sollte folgendermaßen sein:

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
oder ändern
```

2. Error:

```
line 16 print(1)IndentationError: unexpected indent
```

Quick-Help:

Entweder ist die Einrückung der Zeile tatsächlich falsch oder es wurden Tabulator Einrückungen mit Einrückungen mit 4 Leerzeichen kombiniert. Dies wird zurzeit von der Webanwendung nicht unterstützt. In dem initialer Code aller Level wird Tabulator für Einrückungen verwendet.

A.3 Schüler-PDF

Diese Anleitung begleitet dich durch sieben Levels, um die spannende Welt der Programmierung mit Python zu entdecken. Du wirst die Grundlagen dieser mächtigen Programmiersprache kennenlernen, indem du die Turtle (Schildkröte) durch Python-Code auf einem Canvas (Zeichenfläche) steuerst.

Folgende Dinge werden in der Anleitung erklärt (Klicke auf den blauen Text um dorthin zu springen):

1. Wie wird es ablaufen?
2. Programmierung mit Python
3. Turtle Graphics
4. Bedienungsanleitung
5. Programmierkonzepte

Lass uns deine aufregende Reise in die Python-Programmierung starten!

Wie wird es ablaufen?

In der Webanwendung gibt es sieben Level, die euch die Grundlagen der Programmierung in Python näher bringen, sowie Level 0, was ein bisschen vorbereitet, und Level Sandbox, wo man Sachen ausprobieren kann.

Das Ziel für jedes Level ist es, die Aufgaben zu lösen, indem Ihr die Turtle durch Programmierung das erwünschte Ergebnis zeichnen lässt. Jedes Level wird einem ein neue Grundlage gezeigt, die dann angewendet werden muss.

Programmierung mit Python



Was ist eine Programmiersprache?

Eine Programmiersprache ist wie ein Werkzeug für den Menschen, um mit dem Computer zu sprechen. Dieses Werkzeug ist dazu da, um dem Computer zu sagen, was es tun soll.

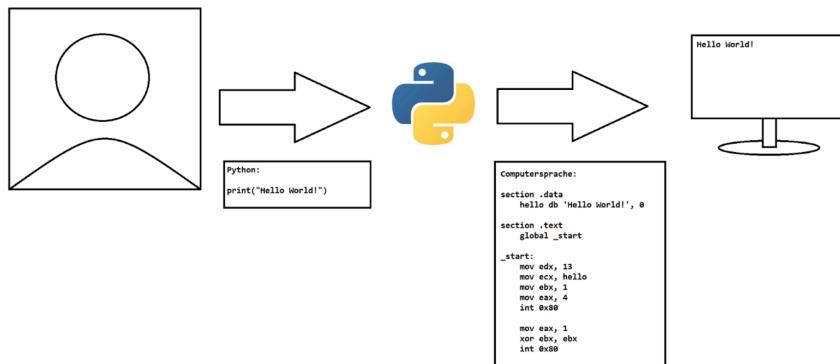
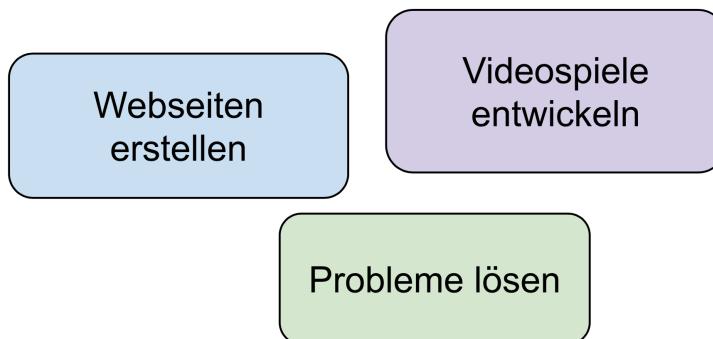


Abbildung 14: Python-Code umwandeln in Computersprache [4]

Wenn man eine Programmiersprache benutzt, gibt man dem Computer Anweisungen, damit verschiedene Aufgaben gelöst werden können, wie z.B.:



Ist die Reihenfolge im Code wichtig?

In der Programmierung ist die Reihenfolge der Anweisungen wichtig, damit am Ende alles funktioniert. Du möchtest ja nicht sagen, er soll zuerst den Text schreiben, und dann das Heft öffnen, da sonst alles übereinander stehend vorne auf dem Heft steht, sondern er soll ja zuerst das Heft öffnen, dann den Text schreiben, umblättern, und den Text zu Ende schreiben.

Es gibt viele verschiedene Programmiersprachen, die ihre eigenen Regeln und Möglichkeiten bieten. Manche Sprachen sind einfacher zu lernen und zu verwenden, andere bieten mehr Möglichkeiten, sind aber vielleicht etwas komplizierter.

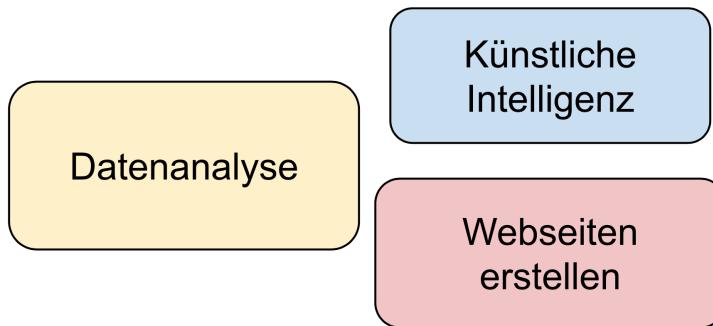
Eine der einfacheren Sprachen ist Python, worum es in diesem Projekt auch geht.



Aber wieso ist Python gut für Anfänger?

Python ist eine Sprache, die gut für Anfänger ist, da sie einfach zu lesen und zu verstehen ist. Die Art und Weise, wie der Code geschrieben wird, ähnelt der englischen Sprache sehr. Es gibt weniger Klammern und Sonderzeichen als in anderen Sprachen, was es einfacher macht, zu verstehen, was der Code macht.

Die Python-Sprache ist eine sehr beliebte Sprache für die vielen Ressourcen, die einem helfen könnten, sein Ziel zu erreichen. Es ist mit Python möglich, eine Vielzahl von Projekten umzusetzen:



Python bietet dem Nutzer auch ein schnelles Erfolgserlebnis, was in anderen Sprachen meist nicht so schnell zu sehen ist.



Gibt es auch Nachteile an Python?

Python kann langsamer sein als andere Programmiersprachen wie C, oder C++. Was für Standardanwendungen kein Problem sein sollte, aber für Anwendungen, die hohe Leistungen erfordern, ein Nachteil sein könnte. Der Speicherplatzverbrauch von Python ist auch höher als bei anderen Sprachen, was bei Geräten mit begrenztem Speicher zu Problemen führen könnte.

Die Grundlagen könnten für den Anfang euch vielleicht zu viel sein, aber durch häufige Wiederholung und aktive Programmierung werden diese euch später leicht fallen. Es ist normal anfangs, von dem vielen neuen Zeug überfordert zu sein. Nehmt euch Zeit und geht die Aufgaben in eurem Tempo durch, um die Inhalte wirklich zu verstehen.

Turtle Graphics

In dieser Webanwendung verwendet ihr Turtle Graphics, unterstützt durch die Python-Bibliothek "turtle". Turtle Graphics ermöglicht es euch, Zeichnungen mithilfe von Befehlen zu erstellen. Die Turtle (Schildkröte) fungiert dabei als euer virtueller Stift.

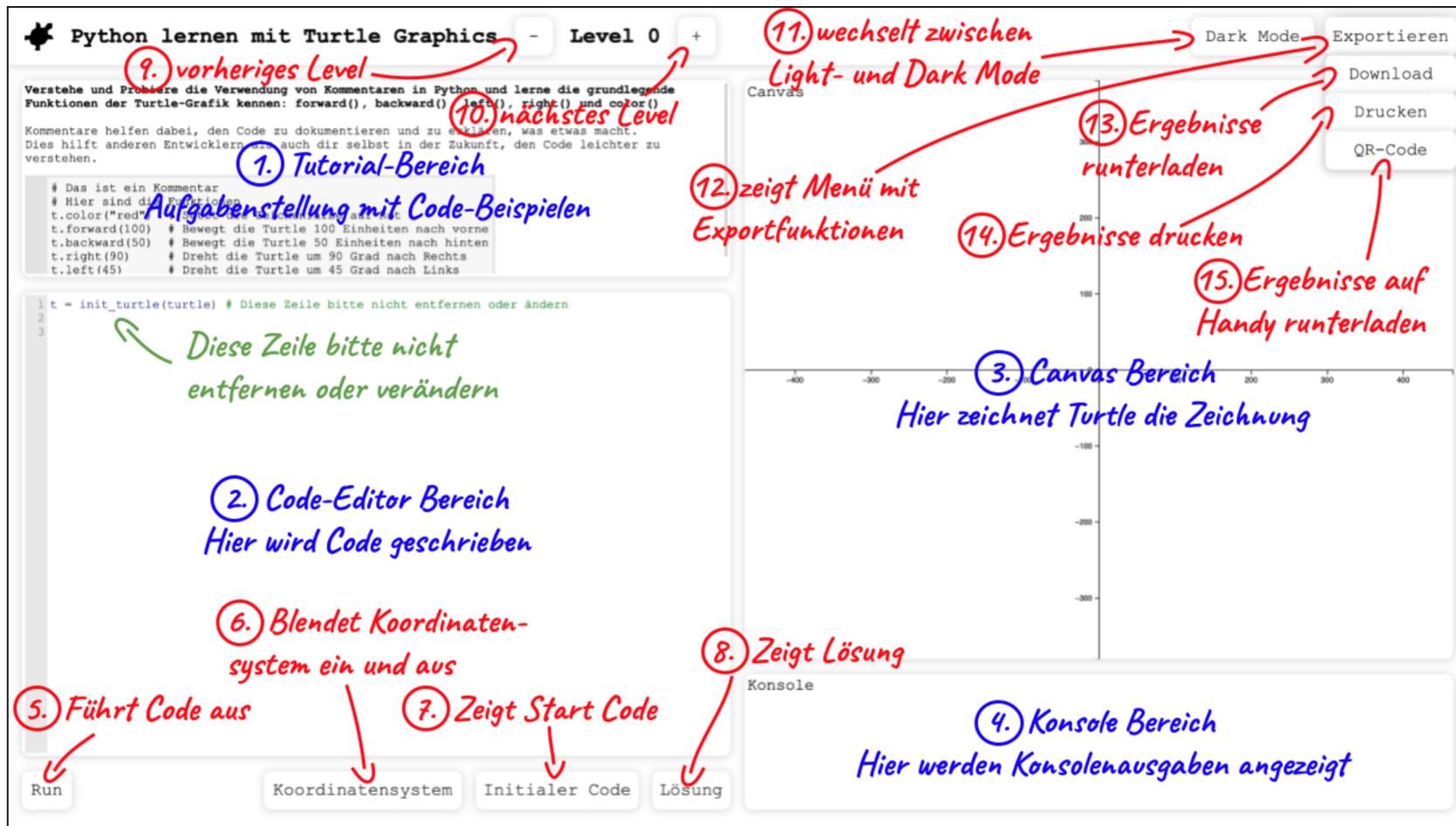
Hier ist eine einfache Tabelle mit einigen grundlegenden Turtle-Befehlen die du gleich verwenden wird. Mach dir keinen Kopf, wenn du beim Lesen verwirrt bist. In jedem Level werden die Funktionen die du im Level brauchst einzeln mit Beispielen erklärt.

Tabelle 2: Turtle Befehle

Befehl	Beschreibung
t.forward(<i>distance</i>)	Bewegt die Turtle vorwärts um die angegebene Distanz.
t.backward(<i>distance</i>)	Bewegt die Turtle rückwärts um die angegebene Distanz.
t.left(<i>angle</i>)	Dreht die Turtle nach links um den angegebenen Winkel (in Grad).
t.right(<i>angle</i>)	Dreht die Turtle nach rechts um den angegebenen Winkel (in Grad).
t.color(<i>color</i>)	Setzt die Farbe für zukünftige Zeichnungen (z.B. ' <i>red</i> ' oder '#00FF00').
t.begin_fill()	Startet das Ausfüllen des Bereichs, der von zukünftigen Zeichnungen umschlossen wird.
t.end_fill()	Beendet das Ausfüllen des Bereichs und füllt ihn mit der ausgewählten Farbe.
t.width(<i>width</i>)	Setzt die Breite des Stifts für zukünftige Zeichnungen.
t.hideturtle()	Blendet die Turtle aus, sodass sie nicht mehr sichtbar ist.
t.showturtle()	Zeigt die Turtle an, wenn sie zuvor ausgeblendet wurde.
t.speed(<i>speed</i>)	Setzt die Zeichengeschwindigkeit der Turtle. Die Geschwindigkeit kann Werte von 0 bis 10 annehmen, wobei 1 die langsamste und 10 die zweit schnellste und 0 die schnellste Geschwindigkeit ist.
t.goto(<i>x, y</i>)	Bewegt die Turtle an die angegebene Position (<i>x, y</i>) im Zeichenfeld.
t.penup()	Hebt den Stift der Turtle an, sodass keine Spur hinterlassen wird.
t.pendown()	Senkt den Stift der Turtle, um eine Spur zu hinterlassen.

Bedienungsanleitung Webanwendung

In der folgenden Darstellung ist die Webanwendung zu sehen. Jedes Level hat eine Aufgabenstellung (1). Code wird im Code-Editor Bereich (2) geschrieben. Sobald der Run-Button (5) gedrückt wird, wird die im Code-Editor definierte Zeichnung auf dem Canvas (3) gezeichnet.



Bereiche (blau)

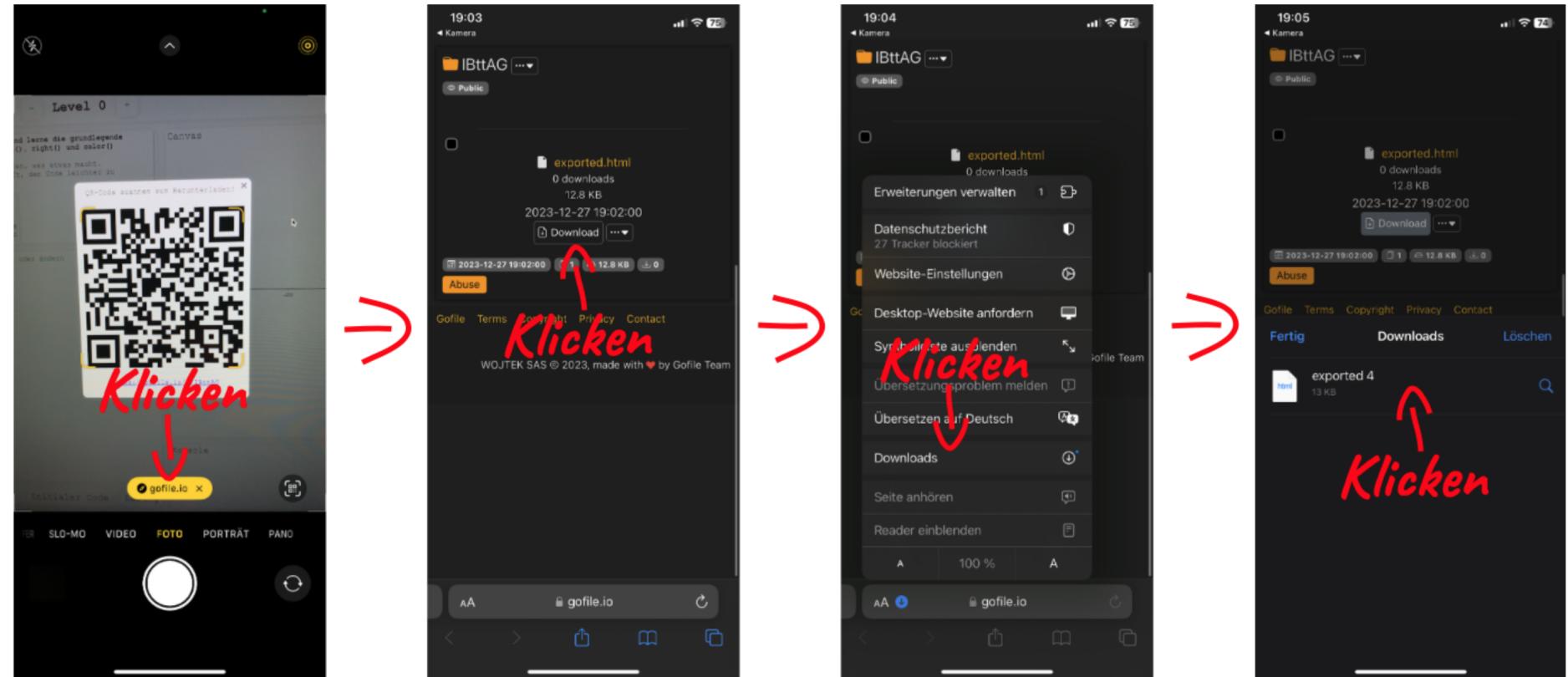
1. **Tutorial Bereich:** Hier steht eine Aufgabenstellung des Levels und Code-Beispiele, die dabei helfen, das Level erfolgreich abzuschließen.
2. **Code-Editor Bereich:** Hier wird Code geschrieben. Hier steht schon zu Beginn für jedes Level ein initialer Code. Jedes Level hat dieselbe erste Zeile, die die Turtle initialisiert. **Die erste Zeile soll nicht entfernt oder verändert werden.** Ansonsten kann es zu Problemen bei der Ausführung kommen.
3. **Canvas Bereich:** Hier zeichnet die Turtle die Zeichnung, sobald der **Run-Button (5)** gedrückt wird.
4. **Konsolen Bereich:** Hier werden Konsolenausgaben angezeigt.

Buttons/Knöpfe (rot)

5. **Run:** Führt Code aus dem **Code-Editor (2)** aus.
6. **Koordinatensystem:** Blendet Koordinatensystem im **Canvas (3)** ein und aus. Beim Zeichnen hilft es, mit Längen und Koordinaten zu zeichnen.
7. **Initialer Code:** Zeigt Start Code und hilft dabei, wenn man sich verrannt hat oder die erste Zeile entfernt wurde.
8. **Lösung:** Zeigt die Lösung in zwei Schritten. Erster Schritt nur Kommentare der Lösung angezeigt. Zweiter Schritt zeigt die Codelösung mit Kommentaren. Für diese gesamte Lösung braucht man ein Passwort. Dieses erhält man vom Dozenten.
9. -: Vorheriges Level
10. +: Nächstes Level
11. **Dark Mode / Light Mode:** Beim Starten der Anwendung werden Systemeinstellungen verwendet, um zu entscheiden, ob Anwendung im Light- oder Dark Mode ist. Das kann über diesen Button nachträglich gewechselt werden.
12. **Exportieren:** Zeigt Menü mit mehreren Buttons zum Exportieren der Ergebnisse. Die exportierte HTML-Datei kann in jedem Browsern auf PC und Smartphone geöffnet werden.
13. **Download:** Lädt die exportierte HTML-Datei herunter und speichert diese im Download Ordner. Aus diesem kann man die Datei auf einen USB Stick kopieren.
14. **Drucken:** Druckt die exportierte HTML-Datei. Nicht wundern: Der Druckvorgang kann erst nach einigen Sekunden gestartet werden.
15. **QR-Code:** Zeigt QR-Code an. Dieser kann mit Smartphone-Kamera gescannt werden. QR-Code leitet auf dem Smartphone auf der Downloadseite weiter. Hier kann die exportierte HTML-Datei heruntergeladen und später heruntergeladen werden.

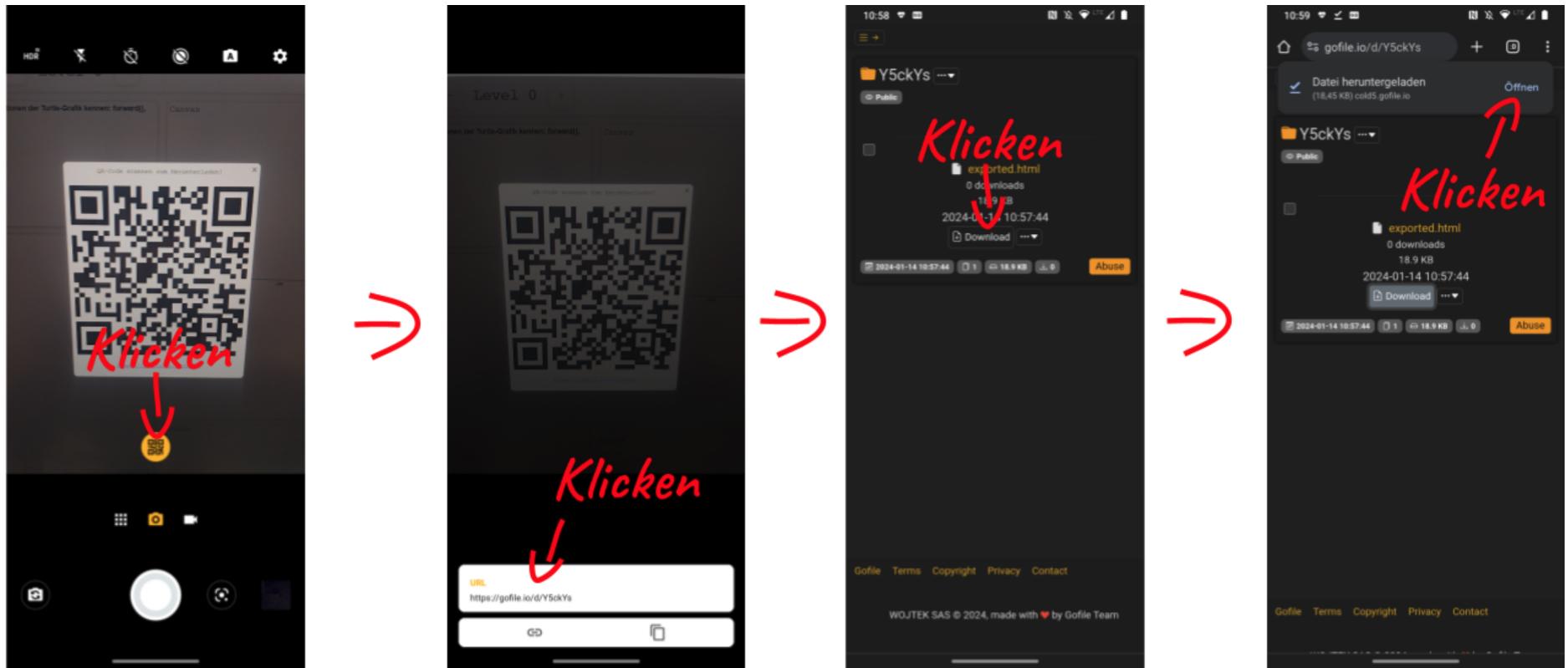
iPhone/iOS

Hier ist dargestellt, welche Schritte man befolgen muss, um die exportierte HTML-Seite auf dem iPhone herunterzuladen und zu öffnen.



Android

Hier ist dargestellt, welche Schritte man befolgen muss, um die exportierte HTML-Seite auf einem Android-Smartphone herunterzuladen und zu öffnen.



Programmierkonzepte

Hier werden Programmierkonzepte erklärt. Diese dienen als Hilfe zum Bearbeiten der Level.

Variablen (Level 1, 2, 3, 4, 5, 6, 7)

Variablen sind wie Container für Informationen in der Programmierung. Du kannst dir eine Variable wie eine Schublade vorstellen, in der du Werte speichern kannst. Diese Werte können Zahlen, Strings (Zeichenfolgen, die Text enthalten) oder andere Daten sein.

Wertzuweisung

Du kannst Variablen einen Namen geben. Über den Namen einer Variable kannst du auf den Wert der Variable zugreifen.



Welchen Wert hat die Variable **b**?

- Richtig, der Wert ist 1, weil der Wert von Variable **a** den Wert auf **b** zugewiesen hat.

Rechnen mit Variablen (Arithmetische Operationen)

Mit Variablen, die Zahlen enthalten, kann gerechnet werden.

```

</> a = 2
      b = 2

      summe = a + b      # Ergebnis einer Addition
      differenz = a - b  # Ergebnis einer Subtraktion
      produkt = a * b    # Ergebnis einer Multiplikation
      quotient = a / b    # Ergebnis einer Division
      rest_der_division = a % b # Liefert Rest einer Division
  
```

← Kennst du noch aus der Grundschule

Ohne zu wissen, dass du es wieder brauchst, hast du den `%` Modulo Operator schon in der Grundschule verwendet.



- Wenn du noch weißt wie in Grundschule geteilt mit Rest gerechnet wurde,
• nenne den Wert von `rest_der_division`?

Richtig, der Wert von `rest_der_division` ist 0.



Hinter einem `#` werden **Kommentare** geschrieben. Kommentare helfen dabei, eigenen und fremden Code zu verstehen.



Wie du siehst, sind alle Variablen vorne kleingeschrieben. Wenn man mehrere Wörter in einem Variablenamen nutzen möchte, dann trennt man diese durch einen Unterstrich. Diesen Code-Stil nennt man **snake_case**. Es gibt auch andere Code-Stile z.B. **camelCase**. Das Programm funktioniert natürlich auch ohne Code-Stil. Aber Informatiker brauchen Regeln, ansonsten gibt es beim Zusammenarbeiten nur Streit.

Schleifen (Level 2, 3, 4, 5, 6, 7)

Schleifen sind einer der Grundbausteine der Programmierung, welche die Programmierung vereinfachen.

Einsatz und Vorteile von Schleifen

Aber wieso sind Schleifen überhaupt so wichtig? Angenommen, du willst 1000-mal das Gleiche schreiben. Das könnte man auf zwei Wege umsetzen.

1. Variante:

```
</> print(0) ← Konsolenausgabe
    print(1)
    print(2)
    print(3)
    print(4)
    print(5)
    print(6)
    print(7)
    print(8)
    print(9)
    # ...
    print(999) # ... 1000 Zeilen später
```

2. Variante:

```
</> for i in range(1000): # [0, 1, 2, ..., 999] ← Array
    print(i) ← startet mit 0
    ↙ eingerückt
```

← Doppelpunkt
Laufvariable



Welche der zwei Varianten ist sinnvoller?

- Ich wette, du hast an Weg 2 gedacht. Informatiker versuchen Code so effizient wie möglich zu schreiben. Das Programmieren geht so nicht nur viel schneller, sondern ist auch viel übersichtlicher und es ist einfacher, etwas am Code zu ändern.

Stell dir vor, du willst statt 0 bis 999 die Zahlen 0 bis 999999 ausgeben. Bei der 1. Variante braucht man eine Million Zeilen. Bei der 2. Variante muss man nur die Zahl n innerhalb von **range (n)** ändern.

Eine For-Schleife hat einen Schleifenkopf, der mit dem Schlüsselwort **for** beginnt. Die Laufvariable **i** (kann auch anders heißen) läuft in diesem Beispiel durch die Werte 0 bis 999 durch.

Für jeden Wert, den **i** dabei annimmt, wird der Schleifenkörper ausgeführt. In dem Schleifenkörper kann die Laufvariable verwendet werden. Außerhalb davon geht das nicht.



Einrückung des Schleifenkörpers

In Python wird der Schleifenkörper mit der Taste TABULATOR eingerückt, um zu zeigen das der eingerückte Code von der **for**-Schleife mehrmals ausgeführt werden soll. Legt man eine Variable innerhalb des Schleifenkörpers an, dann kann man diese wie die Laufvariable nur innerhalb des Schleifenkörpers nutzen.



Beim Programmieren fängt man mit 0 statt mit 1 beim Zählen an. Dadurch wird bei **range(1000)** auch nur bis 999 (1000 - 1) gezählt.



Mit **print()** kann man Werte (auch aus Variablen) auf der Konsole ausgeben.
Mit **print('Text' str(number))** kann man Texte oder Zahlen kombinieren.

Arrays (Level 4, 5)

Ein Array ist eine Datenstruktur in Python und kann mehrere Werte speichern. Das können nicht nur Zahlen, sondern auch alle anderen Werte, wie Strings (Zeichenfolgen) sein. Sogar Arrays können in Arrays gespeichert werden.



Bei der 2. Variante steht hinter **range(1000)** als Kommentar **[0, 1, 2, ..., 999]** um zu zeigen, dass aus **range(1000)** ein Array mit 0 bis 999 generiert wird.

Folgende zwei Codebeispiele zeigen unterschiedlichen Zugriffsmöglichkeiten auf Werte in Arrays. Beide Codes sind beide sinnvoll programmiert und ergeben die gleiche Ausgabe.

Bei einem **direkten** Zugriff wird direkt auf den Wert der Laufvariable **string** zugriffen. Die Laufvariable ändert sich in jedem Durchlauf des Schleifenkörpers.

```
</> string_array = ['eins', 'zwei', 'drei'] ← Array mit Strings
    for string in string_array:
        print(string) ← direkter Zugriff auf Laufvariable
```

Bei einem **indirekter Zugriff mittels Index**, wird mit dem Index `i` auf den Wert auf der Stelle `i` im Array `string_array` zugegriffen.

```
</> string_array = ['eins', 'zwei', 'drei']
    for i in range(3): # [0, 1, 2]
        print(string_array[i])←indirekter Zugriff mittels Index
```

Verschachtelte Schleifen (Level 6)

Schleifen lassen sich verschachteln. Hier ist ein Beispiel:

```
</> for i in range(3): # [0, 1, 2]
    for j in range(4): # [0, 1, 2, 3]
        print(i * j)←zweimal eingerückt
# Ausgabe
# 0, 0, 0, 0, 0, 1, 2, 3, 0, 2, 4, 6
```

If-Bedingung (Level 3, 4, 5, 7)

In der Programmierung gibt es oft Situationen, in denen dein Code Entscheidungen treffen muss. Du kannst es dir so vorstellen, als ob du in deinem Programm verschiedene Pfade vor dir hast und je nach bestimmten Bedingungen einen bestimmten Weg einschlagen musst.

Die **If-Bedingung** sind ein solches Konzept, um diese Situationen zu bewältigen und sicherzustellen, dass dein Code auf verschiedene Szenarien richtig reagieren kann.

```
</> if age < 13: Bedingung
    print('Du bist unter 13 Jahre alt') eingerückt
```

Damit die **If-Bedingung** funktioniert, muss die “Frage” oder Bedingung, die sie stellt, mit “Ja” oder `True` (Wahr) beantwortet werden. Wenn die Antwort “Nein” oder `False` (Falsch) ist, macht das Programm etwas anderes oder geht einfach weiter.

```
</> if age < 13: 1. If-Bedingung
    print('Du bist unter 13 Jahre alt')

if True: 2. If-Bedingung
    print('Dies wird immer ausgeführt')

if False: 3. If-Bedingung
    print('Dies wird nie ausgeführt')
```

Hier überprüft die erste **If-Bedingung**, ob du jünger als 13 Jahre bist. Wenn das stimmt, sagt der Code “Du bist unter 13 Jahre alt”.

Die zweite **If-Bedingung** ist ein bisschen wie ein Scherz. Er sagt immer “Dies wird immer ausgeführt”, egal wie alt du bist. Das liegt daran, dass `True` so viel wie “immer wahr” bedeutet. Deswegen wird dieser Teil immer passieren.

Der letzte Teil ist genau das Gegenteil. Er hat `False` drin, was so viel wie “Immer falsch” bedeutet. Deshalb wird der Satz “Dies wird nie ausgeführt” auch wirklich nie gesagt.

Erweiterte If-Bedingungen

```
</> if age < 13:  
    print('Du bist unter 13 Jahre alt')  
elif age >= 13 and age < 18:  
    print('Du bist ein Teenager, aber nicht erwachsen')  
elif 18 <= age < 20:  
    print('Du bist ein Teenager und erwachsen')  
else:  
    print('Du bist erwachsen')
```



Wenn du 15 Jahre alt bist, was für eine Ausgabe wird ausgegeben?

- Richtig, die Ausgabe ist '**Du bist ein Teenager, aber nicht erwachsen**', da 15 größer gleich 13, aber kleiner als 18 ist.

Um eine **If-Bedingung** zu erweitern, kannst du mit `elif` (eine Abkürzung für "`else if`", was "sonst wenn" bedeutet) eine weitere **If-Bedingung** hinzufügen, die geprüft wird, wenn die vorherige Wahl nicht wahr war.

Mit `else` (also "sonst") bestimmst du, was passieren soll, wenn keine der vorherigen Wahlen zutrifft. So kannst du sicherstellen, dass immer etwas ausgeführt wird, egal ob Bedingungen erfüllt werden können oder nicht.



Bei den If-Bedingungen können auch `and` und `or` verwendet werden, um mehr Szenarien zu berücksichtigen. Bei `and` müssen alle Bedingungen wahr sein, während bei `or` nur einer der Bedingungen wahr sein muss.

Funktionen (Level 5, 6, 7)

Funktionen in der Programmierung sind wie Wege, die du in deinem Code festlegst. Stell dir vor, du hast einen Rucksack voller Werkzeuge (**Funktionen**), die du immer wieder verwenden kannst, ohne sie neu zu packen. Diese Werkzeuge helfen dir, bestimmte Aufgaben wie die Addition von zwei Zahlen schnell und effizient zu erledigen.

Genau wie bei Weggabelungen in einem Programm, wo du mit **If-Bedingungen** entscheiden musst, welchen Pfad du einschlagen willst, rufst du eine **Funktion** auf, wenn du sie benötigst. Diese Entscheidungspunkte stellen sicher, dass dein Code flexibel auf verschiedene Situationen reagieren kann.

Damit man eine **Funktion** erstellt, muss diese vorher definiert werden, dies tut man mit **def**, anschließend den **Namen der Funktion** und die erwünschten, wenn gebrauchten, **Übergabeparameter**.

```
</>          ↗ Funktion
      def funktionsname(a,b):    ↗ Übergabeparameter
          print(a, 'und', b)     ↗ eingerückt
          return a               ↗ Rückgabewert
```

Wenn keine **Übergabeparameter** benötigt werden, können diese entfernt werden. Die Klammern müssen aber, vollständigkeitshalber, vorhanden sein.

```
</> def funktion_2():
      return 'Rückgabe'

print(funktion_2())
```



Funktionsnamen, Duplikate und Schlüsselwörter

Funktionen brauchen einen eigenen Namen, der nicht von anderen Funktionen, sowie von Schlüsselwörtern wie “**for**, **if**, **elif**,...”, verwendet wird.

Um eine Funktion zu verwenden, muss diese vorher aufgerufen werden.

```
</> def addition(x,y):
    return (x+y)

print(addition(1.2,2))
print(addition(15,-2))
print(addition(0,56))
print(addition((3+4),56))
```

Man braucht nicht mit `return` etwas zurückzugeben. Manche Funktionen führen nur etwas aus und müssen nichts zurückzugeben.

Falls man aber keine Werte für einen der Übergabeparameter hat, kann man in der Funktion selbst einen Standardwert vordefinieren. Dies macht es, dass selbst wenn kein Parameter übergeben wird, die Funktion aufrufbar ist.

Aber was ist, wenn man nur auf `y` zugreifen möchte. Dann kann man in der Übergabe dies definieren, indem man den direkten Variablennamen der Funktion "überschreibt".

```
</> def subtraktion(x=3,y=5):
    return (x-y)

print(subtraktion()) # Ausgabe: -2
print(subtraktion(x=17)) # Ausgabe: 12
print(subtraktion(y=2)) # Ausgabe: 1
print(subtraktion(y=2, x=17)) # Ausgabe: 15
print(subtraktion(x=15, y=-17)) # Ausgabe: 32
```



Was ist die Ausgabe der Subtraktions-Funktion, wenn keine Übergabeparameter übergeben werden?

Richtig, die Ausgabe ist **-2**, da `x=3` und `y=5` ist und somit $x - y = 3 - 5 = -2$ ist.

In den vorherigen Kapiteln haben wir schon die `print()` Funktion benutzt, die Texte in der Konsole anzeigt. Normalerweise macht `print()` am Ende einen Zeilenumbruch "`\n`", aber mit dem `end` Parameter kannst du das ändern und entscheiden, was am Ende stehen soll.

```
</> print('Das ist der Übergabeparameter', end='Ende')
```

B Anhang - Programmieren mit Audio

B.1 Lehrer-PDF

B.1.1 Installation

Folgen Sie diesen Schritten um das Projekt zu installieren und den Webeditor aufzurufen:

- Stellen Sie sicher, das auf dem Zielsystem Python installiert ist.
- Rufen Sie die Gitlab Seite der Projektgruppe auf (<https://gitlab.technik-emden.de/da6338/projektgruppe-23-link>).
- Wählen Sie den Branch `musik-soundsäus`.
- Drücken Sie auf den blauen Button `Code` und wählen unter "Download source code" das gewünschte Kompressionsformat aus. Es sollte der Download des source codes beginnen.
- Nach dem Download entpacken Sie die Datei an einen gewünschten Zielort.
- Öffnen Sie den Ordner `projektgruppe-23-link-musik-sounds` und folgen Sie dem Pfad `projektgruppe-23-link-musik-sounds` → `Programmieren mit Audio` → `SSource-Code`. Sie sollten jetzt in einem Verzeichnis sein, das unter anderem eine HTML-Datei namens `code-editor.html` sowie ein Verzeichnis namens `levels` enthalten sollte.
- Öffnen Sie in diesem Ordner ein Terminal. Gegebenenfalls können Sie nicht direkt im Ordner ein Terminal öffnen, Sie sollten dann ein allgemeines Terminal Fenster öffnen und in den Ordner namens `SSource-Codes` navigieren.
- Sind Sie im Ordner angekommen, geben Sie `python -m http.server 8080` ein um den Server des Webeditors auf Port 8080 zu öffnen.
- Öffnen Sie jetzt einen beliebigen Browser und geben Sie in die Adressleiste `http://localhost:8080/code-editor.html`
- Es sollte sich der Webeditor des Projekts öffnen. Ab jetzt können die Aufgaben bearbeitet werden.

B.1.2 Musterlösungen

B.1.2.1 Musterlösung Aufgabe 0 - Variablen, Arithmetik und das Ausgeben auf der Konsole

Erstellen Sie eine Variable a und eine Variable b, a soll den Wert 10 haben und b den Wert 23. Erstellen Sie zudem noch eine Variable c, die als Wert das Ergebnis der Addition von a und b enthält und geben Sie diese auf der Konsole aus.

Listing 19: Lösung: Aufgabe 0

```
a = 10  
b = 23  
c = a + b
```

```
print(c)
```

```
Konsole  
33
```

B.1.2.2 Musterlösung Aufgabe 1 - If/Else Verzweigung

Erstellen Sie eine Variable a mit dem Wert 10. Schreiben Sie jetzt eine If/Else Verzweigung, die überprüft, ob a kleiner/gleich 10 ist oder ob a größer als 10 ist. Ist a größer, dann lassen Sie "Größeräuf der Konsole ausgeben, ist a kleiner oder gleich 10, dann lassen Sie "Kleineräuf der Konsole ausgeben (auf der Konsole soll hierbei der Wert, den Sie a gegeben haben, ausgegeben werden). Testen Sie ihr Programm, indem Sie den Wert der Variable a verändern und das Programm erneut starten.

Listing 20: Lösung: Aufgabe 1

```
a = 9
if(a <= 10):
    print("Kleiner")
else:
    print("Groesser")
```

Konsole
Kleiner

```
a = 10
if(a <= 10):
    print("Kleiner")
else:
    print("Groesser")
```

Konsole
Kleiner

```
a = 11
if(a <= 10):
    print("Kleiner")
else:
    print("Groesser")
```

Konsole
Groesser

B.1.2.3 Musterlösung Aufgabe 2 - Schleifen

Erstellen Sie eine Variable a mit dem Wert 0. Danach erstellen Sie eine for oder while Schleife, die von 0 bis 10 geht. Bei jedem Schleifendurchlauf soll der Wert des aktuellen Durchlaufs auf a oben drauf addiert werden. Wenn die Schleife durchgelaufen ist, soll a ausgegeben werden. Als Tipp: Bei einer Schleife mit vier Durchläufen (0-3) wäre der Wert der Variable a am Ende 6.

```
Listing 21: Lösung: Aufgabe 0
a = 0
for x in range(0, 11):
    a = a + x

print(a)

Konsole
55
```

B.1.2.4 Musterlösung Aufgabe 3 - Töne abspielen

Erstellen Sie wie in Aufgabe 1 eine Variable a mit dem Wert 10 und eine Verzweigung, die überprüft, ob a kleiner/gleich oder größer 10 ist. Diesmal soll aber nichts auf der Konsole ausgegeben werden. Wenn a kleiner/gleich 10 ist, soll mithilfe der tone() Funktion der Ton C ausgegeben werden, ist a größer als 10 soll der Ton "D" ausgegeben werden. Überprüfen Sie ihren Code wie in Aufgaben 1, indem Sie den Wert der Variable a verändern und das Programm erneut starten.

```
Listing 22: Lösung: Aufgabe 0
a = 9
if(a <= 10):
    tone("C")
else:
    tone("D")

a = 10
if(a <= 10):
    tone("C")
else:
    tone("D")

a = 11
if(a <= 10):
    tone("C")
else:
    tone("D")
```

B.1.2.5 Musterlösung Aufgabe 4 - C-Dur-Tonleiter erstellen

Sie haben folgendes Array gegeben: töne = [C", "C", "E", "D", F", "E", "G", F", C", "G", "H", Ä", "H", "H", C", C"]. Erstellen Sie jetzt ein leeres Array namens C_Dur_Tonleiter. Benutzen Sie eine Schleife und eine If/Else Verzweigung, um das Array so zu füllen, das es eine C-Dur-Tonleiter (C, D, E, F, G, A, H, C) darstellt und lassen Sie es mithilfe der tones() Funktion abspielen. Kleiner Tipp: Sie müssen jedes zweite Element aus dem töne Array in das C-Dur-Tonleiter Array eintragen lassen. (Beim Abspielen von Ton "H" kommt es zu einem Fehlerton).

```
Listing 23: Lösung: Aufgabe 0
toene = ["C", "E", "D", "F", "E", "H", "F", "G",
          "G", "E", "A", "C", "H", "C", "C"]
C_Dur_Tonleiter = []

for x in range(0, len(toene)):
    if(x % 2 == 0):
        C_Dur_Tonleiter.append(toene[x])

tones(C_Dur_Tonleiter)
```

B.1.2.6 Musterlösung Aufgabe 5 - ABC Notation

Gehen Sie auf diese [Seite](#) und suchen Sie sich eine ABC-Notation heraus. Kopieren Sie die Notation in den Editor und wandeln Sie sie so um das die play() Funktion sie abspielen kann und lassen Sie die Notation abspielen.

```
Listing 24: Lösung: Aufgabe 0
ABC Notation
(Speed the Plough, Startseite(https://abcnotation.com/) ):

X:1
T:Speed the Plough
M:4/4
C:Trad.
K:G
| :GABC dedB|dedB dedB|c2ec B2dB|c2A2 A2BA |
    GABC dedB|dedB dedB|c2ec B2dB|A2F2 G4:|
| :g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df |
    g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|
play("X:1 \n"+
  "T:Speed the Plough \n"+
  "M:4/4 \n"+
  "C:Trad. \n"+
  "K:G \n"+
  "| :GABC dedB|dedB dedB|c2ec B2dB|c2A2 A2BA| \n"+
  "| GABC dedB|dedB dedB|c2ec B2dB|A2F2 G4:| \n"+
  "| :g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df| \n"+
  "| g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|")
```

B.2 Schüler-PDF

B.2.1 Lernziele

Ziel dieser Übung ist es, erste allgemeine Einblicke in die Programmierung zu bekommen. Sie sollen hierbei aber nicht nur irgendwelche Buchstaben und Zahlen auf der Konsole ausgeben, sondern im Verlaufe der Übung auch Töne und Musik abspielen lassen.

Nach der Bearbeitung sollten Sie folgende Kenntnisse erlangt haben:

- Sie können Dinge auf der Konsole ausgeben lassen.
- Sie kennen die grundlegenden Bausteine, mit denen man den Ablauf eines Programms steuern kann.
- Sie können mithilfe der `tone()` und `tones()` Funktionen beliebig viele Töne abspielen lassen.
- Sie können eine ABC-Notation so verändern, das sie von der `play()` Funktion abgespielt werden kann.

B.2.2 Variablen

In Variablen werden Informationen abgespeichert.

```
Listing 25: Variablen in Python
a = 10 a wird der Wert 10 zugewiesen.
b = 230 b wird der Wert 230 zugewiesen.
c = a c wird der Wert von a zugewiesen.
wort = 'Hallo' wort wird der Text Hallo zugewiesen.
```

Variablen, die ganze Zahlen speichern, heißen integer und Variablen, die einzelne Buchstaben oder ganze Texte speichern, heißen Strings. Bei Strings ist zu beachten, dass sie mit am Anfang und am Ende mit einem ' oder einem " gekennzeichnet werden.

B.2.2.1 Rechnen mit Variablen

Enthalten Variablen Zahlen, dann kann mit ihnen gerechnet werden.

```
Listing 26: Rechnen mit Variablen
a = 180
b = 90

summe = a + b -> Addition
differenz = a - b -> Subtraktion
produkt = a * b -> Multiplikation
quotient = a / b -> Division
divisionsrest = a % b -> Der Rest nach einer Division
```

B.2.2.2 Ausgeben von Variablen

Will man Variablen in der Konsole ausgeben lassen, geht das wie folgt:

```
Listing 27: Ausgeben von Variablen
a = 120
b = 80
c = a + b

print(a) -> gibt 120 auf der Konsole aus
print(c) -> gibt 200 auf der Konsole aus
print(a + b) -> gibt auch 200 auf der Konsole aus
```

B.2.3 Schleifen

Schleifen sind Codeblöcke, die man beliebig oft wiederholen lassen kann. Obwohl es mehrere Schleifen Varianten gibt, werden wir uns hier nur auf die for Schleife konzentrieren.

Der folgende Code beschreibt eine for Schleife, die von 0 bis 10 hochzählt und die jeweilige Zahl auf der Konsole ausgibt.

Listing 28: Aufbau der for Schleife

```
for x in range(0, 11):
    print(x)
```

Listing 29: Konsolenausgabe

```
0
1
2
3
4
5
6
7
8
9
10
```

Wichtig ist zu beachten, dass die hintere Zahl (im Beispiel 11) immer einen größer sein muss als die Zahl, zu der man zählen will.

B.2.4 Arrays

Ein Array ist eine Liste, die Dinge wie Zahlen oder Texte abspeichern kann.

```
Listing 30: Gemischte Arrays
integer_array = [1, 2, 3, 4, 5]
string_array = ['eins', 'zwei', 'drei']
gemischtes_array = [1, 'zwei', 'C', 'X']
```

Man kann auf ein Array auch zugreifen oder es verändern. Dabei muss man aber beachten, das Arrays nicht bei 1 anfangen, sondern bei 0.

```
Listing 31: Arbeiten mit Arrays
liste = [23, 83, 24, 35, 97]
        0 1 2 3 4

a = liste[2] -> a bekommt den Wert 24 zugewiesen
liste.append(100) -> Die Zahl 100 wird angehaengt
liste[0] = 10 -> Wert an Stelle 0(23) wird zu 10
del liste[0] -> Wert an der Stelle 0 wird geloescht
```

Will man ein Array mithilfe einer for Schleife durchgehen und die Werte der Liste ausgeben, geht das wie folgt:

```
Listing 32: Werte eines Arrays ausgeben
liste = [23, 83, 24, 35, 97]
        0 1 2 3 4

for x in range(0, len(liste)):
    print(liste[x])
```

Konsolenausgabe:

```
23
83
24
35
97
```

Hierbei verwendet man für die hintere Zahl die len() Funktion. Man übergibt der Funktion die Liste und man bekommt die Länge der Liste als integer zurück (mehr zu Funktionen später).

B.2.5 If/Else

Muss innerhalb vom Code eine Entscheidung getroffen werden, bildet man das oft mit einer If/Else Verzweigung ab. Je nachdem wie die Ausgangssituation ist, wird dann entschieden, wie es weitergehen soll.

Als Beispiel wollen wir herausfinden, ob die Variable a größer als 20 ist.

```
Listing 33: a größer 20
a = 10
if a > 20:
    print('a ist grösser als 20')
```

Dazu wollen wir jetzt aber auch etwas ausgeben, wenn a kleiner als 20 ist.

```
Listing 34: a größer/kleiner 20
a = 10
if a > 20:
    print('a ist grösser als 20')
else:
    print('a ist kleiner als 20')
```

Wollen wir jetzt auch noch etwas spezielles ausgeben, wenn a genau 20 ist, geht das auch.

```
Listing 35: a größer/kleiner/gleich 20
a = 10
if a > 20:
    print('a ist grösser als 20')
elif a == 20:
    print('a ist genau 20')
else:
    print('a ist kleiner als 20')
```

B.2.6 Funktionen

Auch wenn wir Funktionen in dieser Übung eher indirekt benutzen, sollten Sie einen groben Überblick darüber bekommen, was sie sind und wie sie funktionieren.

Funktionen sind Codeblöcke, die Sie in ihrem Code einmal definieren und dann an verschiedenen Stellen immer wieder aufrufen und verwenden können.

Eine Funktion hat den allgemeinen Aufbau:

```
Listing 36: Aufbau einer Funktion
def funktionsname(eingabeparameter 1, ...):
    code....
    return rueckgabeparameter 1, ...
```

Zu beachten ist, dass eine Funktion beliebig viele Eingab- und Rückgabeparameter haben kann, das bedeutet, dass sie auch komplett ohne auskommt.

Als Beispiel betrachten wir eine Funktion, die 3 Eingabeparameter addiert und das Ergebnis zurückgibt.

```
Listing 37: Funktionsbeispiel
def summe_aus_drei(a, b, c):
    ergebnis = a + b + c
    return ergebnis

x = 10
y = 20
z = 30
summe = summe_aus_drei(x, y, z)
print(summe) -> auf der Konsole wird 60 ausgegeben
```

Es ist zu beachten, dass die Funktion erst nach ihrer eigenen Definition aufgerufen werden kann.

B.2.7 Musik und Töne

Der Webeditor benutzt im Hintergrund die Bibliothek `musical.js` um die Töne sowie die [ABC Notation](#) abzuspielen.

B.2.7.1 Die tone() Funktion

Mithilfe der `tone()` Funktion lässt sich ein Ton oder mehrere Töne gleichzeitig abspielen.

```
Listing 38: Einzelnen Ton abspielen
tone('C')
```

```
Listing 39: Mehrere Töne gleichzeitig abspielen
tone('C')
tone('D')
tone('C')
```

B.2.7.2 Die tones() Funktion

Mithilfe der `tones()` Funktion lassen sich Töne nacheinander abspielen. Dafür müssen die Töne allerdings zu erst in ein Array abgespeichert werden.

```
Listing 40: Töne nacheinander abspielen lassen
toene = ['C', 'D', 'E']
tones(toene)
```

B.2.7.3 Die play() Funktion

Mithilfe der play() Funktion lässt sich eine beliebige ABC-Notation abspielen, vorausgesetzt man verändert die Notation so, das sie als String dargestellt wird.

Als Beispiel nutzen wir die Notation von Speed the Plough zu finden, auf der dieser [Seite](#).

```
Listing 41: ABC Notation
X:1
T:Speed the Plough
M:4/4
C:Trad.
K:G
| :GABC dedB|dedB dedB|c2ec B2dB|c2A2 A2BA |
    GABC dedB|dedB dedB|c2ec B2dB|A2F2 G4:|
| :g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df |
    g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|
```

```
Listing 42: ABC Notation als String
"X:1 \n"+
"T:Speed the Plough \n"+
"M:4/4 \n"+
"C:Trad. \n"+
"K:G \n"+
" | :GABC dedB|dedB dedB|c2ec B2dB|c2A2 A2BA| \n"+
" GABC dedB|dedB dedB|c2ec B2dB|A2F2 G4:| \n"+
" | :g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df| \n"+
" g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|"
```

```
Listing 43: ABC Notation mit play() abspielen lassen
play(
"X:1 \n"+
"T:Speed the Plough \n"+
"M:4/4 \n"+
"C:Trad. \n"+
"K:G \n"+
" | :GABC dedB|dedB dedB|c2ec B2dB|c2A2 A2BA| \n"+
" GABC dedB|dedB dedB|c2ec B2dB|A2F2 G4:| \n"+
" | :g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df| \n"+
" g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|")
```

B.2.8 Aufgaben

Sie finden die Aufgaben/Levels auch im Webeditor.

B.2.8.1 Aufgabe 0 - Variablen, Arithmetik und das Ausgeben auf der Konsole

Erstellen Sie eine Variable a und eine Variable b, a soll den Wert 10 haben und b den Wert 23. Erstellen Sie zudem noch eine Variable c, die als Wert das Ergebnis der Addition von a und b enthält und geben Sie diese auf der Konsole aus.

B.2.8.2 Aufgabe 1 - If/Else Verzweigung

Erstellen Sie eine Variable a mit dem Wert 10. Schreiben Sie jetzt eine If/Else Verzweigung, die überprüft, ob a kleiner/gleich 10 ist oder ob a größer als 10 ist. Ist a größer, dann lassen Sie "Größer auf der Konsole ausgeben, ist a kleiner oder gleich 10, dann lassen Sie "Kleiner auf der Konsole ausgeben (auf der Konsole soll hierbei der Wert, den Sie a gegeben haben, ausgegeben werden). Testen Sie ihr Programm, indem Sie den Wert der Variable a verändern und das Programm erneut starten.

B.2.8.3 Aufgabe 2 - Schleifen

Erstellen Sie eine Variable a mit dem Wert 0. Danach erstellen Sie eine for oder while Schleife, die von 0 bis 10 geht. Bei jedem Schleifendurchlauf soll der Wert des aktuellen Durchlaufs auf a oben drauf addiert werden. Wenn die Schleife durchgelaufen ist, soll a ausgegeben werden. Als Tipp: Bei einer Schleife mit vier Durchläufen (0-3) wäre der Wert der Variable a am Ende 6.

B.2.8.4 Aufgabe 3 - Töne abspielen

Erstellen Sie wie in Aufgabe 1 eine Variable a mit dem Wert 10 und eine Verzweigung, die überprüft, ob a kleiner/gleich oder größer 10 ist. Diesmal soll aber nichts auf der Konsole ausgegeben werden. Wenn a kleiner/gleich 10 ist, soll mithilfe der tone() Funktion der Ton C ausgegeben werden, ist a größer als 10 soll der Ton "D" ausgegeben werden. Überprüfen Sie ihren Code wie in Aufgaben 1, indem Sie den Wert der Variable a verändern und das Programm erneut starten.

B.2.8.5 Aufgabe 4 - C-Dur-Tonleiter erstellen

Sie haben folgendes Array gegeben: töne = [C", C", "E", "D", F", "E", "G", F", C", "G", "H", "A", "H", "H", C", C"]. Erstellen Sie jetzt ein leeres Array namens C_Dur_Tonleiter. Benutzen Sie eine Schleife und eine If/Else Verzweigung, um das Array so zu füllen, das es eine C-Dur-Tonleiter (C, D, E, F, G, A, H, C) darstellt und lassen Sie es mithilfe der tones() Funktion abspielen. Kleiner Tipp: Sie müssen jedes zweite Element aus dem töne Array in das C-Dur-Tonleiter Array eintragen lassen. (Beim Abspielen von Ton "H" kommt es zu einem Fehlerton).

B.2.8.6 Aufgabe 5 - ABC Notation

Gehen Sie auf diese [Seite](#) und suchen Sie sich eine ABC-Notation heraus. Kopieren Sie die Notation in den Editor und wandeln Sie sie so um das die play() Funktion sie abspielen kann und lassen Sie die Notation abspielen.

C Anhang - CodeClash

C.1 Lehrer-PDF

C.1.1 Inhaltsverzeichnis

- Installation für Mac Geräte
- Lernziele
- Benutzeroberfläche
- Blöcke
- Aufgabenbeschreibung
- Lösungen

C.1.2 Installation für Mac-Geräte

1. Virtualbox Version 7.0.12 installieren.
2. Ubuntu Version 23.04 VM in Virtualbox aufsetzen.
3. Linux-Export des Spiels in Ubuntu herunterladen.
4. In Ubuntu das Terminal im Zielordner öffnen.
5. Das Spiel mit dem Befehl: `chmod +x CodeClash.exe.sh` ausführbar machen.
6. Mit dem Befehl `./CodeClash.exe.sh` ausführen.

C.1.3 Lernziele

Die Schüler sollen mithilfe dieses Spiels die grundlegenden Funktionen der Programmierung kennenlernen.

Dabei soll das Spiel ermöglichen, dass Spieler ohne programmiererische Vorkenntnisse Programme entwerfen können.

Der Spieler muss sich dabei keine Gedanken um die Programmiersprache, IDE oder die zugrunde liegende Syntax machen.

Der Spieler lernt die Benutzung von Bedingungen und Schleifen zur Lösung gestellter Aufgaben.

C.1.4 Benutzeroberfläche:

- Leiste mit Blöcken am unteren Bildschirmrand: Diese Blöcke müssen platziert und teilweise kombiniert werden, um ein Programm zu erstellen und die Aufgabe zu bewältigen.
- Blöcke: Diese Blöcke können per Drag'n'Drop auf der "Platine" platziert werden. Sie beinhalten immer eine Funktion einer spezifischen Programmstruktur.

- Viereckige Slots auf der “Platine”: Hier werden die Blöcke platziert. Die Linien zwischen den Slots dienen danach als Verknüpfung der unterschiedlichen Blöcke.
- **DELETE-Button:** Mit diesem Button kann der aktuelle Levelfortschritt zurückgesetzt und es kann von vorne begonnen werden, wenn man einen Fehler gemacht hat.
- **START-Button:** Sind alle Blöcke platziert, die man platzieren wollte, kann dieser Button gedrückt werden, um das Programm zu starten. Nun wird geprüft, ob das Programm funktionsfähig ist und die Aufgabe somit erfolgreich bearbeitet wurde.
- **HELP-Button:** Mit diesem Button lässt sich ein Pop-Up öffnen, das Hilfestellungen zur jeweiligen Aufgabe bietet.
- **BACK-Button:** Mit diesem Button kommt man zurück zur vorherigen Ansicht (Menüs).

C.1.5 Blöcke:

- **START-Block:** Dieser Block ist statisch und symbolisiert den Programmstart.
- **ZIEL-Block:** Dieser Block ist ebenfalls statisch und symbolisiert das Ende des Programms.
- **WHILE-Block:** Dieser Block erhöht eine Variable (hier: x) so lange, bis der im Block angegebene Wert erreicht wurde und leitet das Signal dann zum nächsten Block weiter.
- **IF-Block:** Dieser Block prüft, ob eine Variable (hier: x) einen gegebenen Wert hat oder nicht, je nachdem wird entweder der rote (false) oder der grüne (true) Ausgang verwendet.
- **END-Block:** Dieser Block wird symbolisch für den nicht verwendeten Ausgang eines IF-Blocks verwendet (hier meist “false”).

C.1.6 Aufgabenbeschreibung

Das Spielfeld besitzt einen unbeweglichen Start- und Zielblock, sowie Felder auf denen Blöcke platziert werden können.

Der Startblock symbolisiert den Beginn des Programms und der Zielblock das Ende des Programms.

Die Felder werden als Platinen dargestellt und die Verbindung der Felder sind als Leiterbahnen dargestellt.

Der Spieler zieht per Drag and Drop die gewünschten Blöcke aus der unteren Leiste auf eines der Felder.

Um zu prüfen, ob das Programm die gestellten Anforderungen erfüllt drückt der Spieler auf den Start-Knopf in der oberen Leiste. Sollte die Lösung des Spielers falsch sein oder der Spieler unzufrieden mit seiner Auswahl sein kann der Delete-Button gedrückt werden um das Level zurückzusetzen.

Das dritte Level baut auf den ersten beiden Leveln auf. Dem Spieler wird daher empfohlen die Level in der richtigen Reihenfolge zu absolvieren.

C.1.7 Lösungen

Ist ein Level korrekt gelöst wird dies dem Spieler in der oberen Leiste angezeigt. In der Level-Übersicht verändert sich die Farbe des Level-Buttons, sollte das Level einmal korrekt gelöst worden sein.

Level-Knopf solange das Level noch nicht gelöst wurde:



Level-Knopf sobald das Level gelöst wurde:



C.1.8 Level 1:

Lösung in [Abbildung 15.](#)

C.1.9 Level 2:

Lösung in [Abbildung 16.](#)

C.1.10 Level 3:

Lösung in [Abbildung 17.](#)

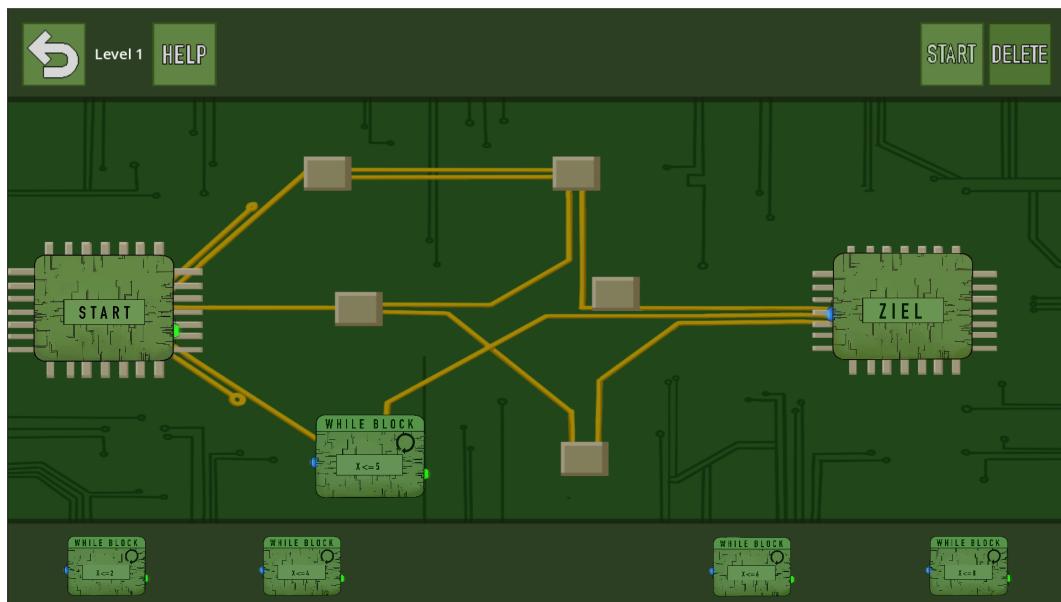


Abbildung 15: Lösung Level 1

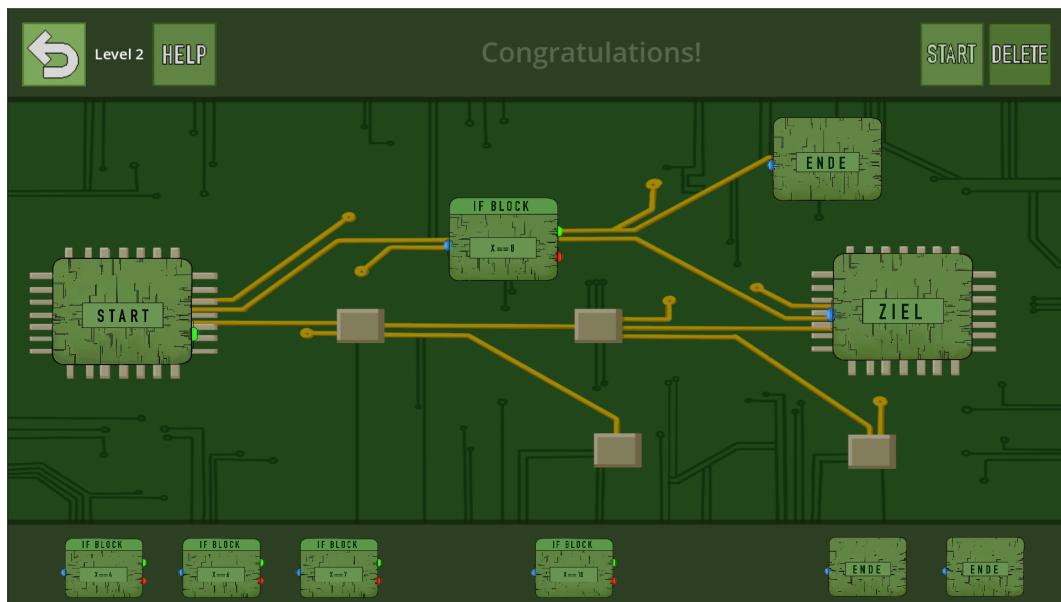


Abbildung 16: Lösung Level 2

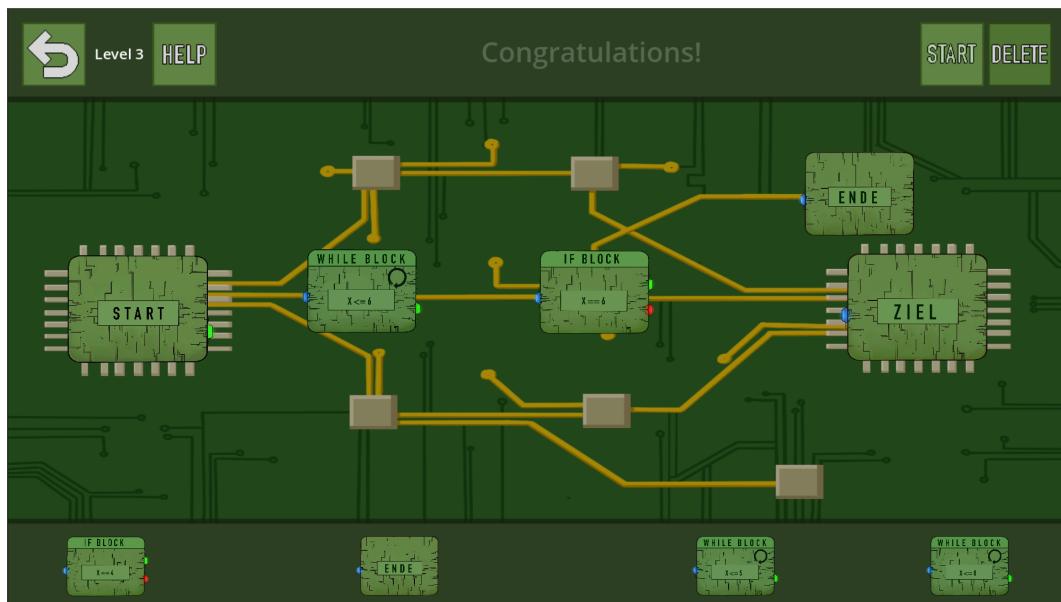


Abbildung 17: Lösung Level 3

C.2 Schülerhandbuch: CodeClash

C.2.1 Ziele:

- Grundlegende Einführung in typische Funktionen der Programmierung
- Spielerisches Lernen von Programmstrukturen und deren Logik

C.2.2 Einführung:

CodeClash zeigt euch spielerisch wie Programme strukturell aufgebaut sind und wie verschiedene Bausteine eines Programms funktionieren.

C.2.3 Benutzeroberfläche:

- Leiste mit Blöcken am unteren Bildschirmrand: Diese Blöcke müssen platziert und teilweise kombiniert werden, um ein Programm zu erstellen und die Aufgabe zu bewältigen.
- Blöcke: Diese Blöcke können per Drag'n'Drop auf der "Platine" platziert werden. Sie beinhalten immer eine Funktion einer spezifischen Programmstruktur.
- Viereckige Slots auf der "Platine": Hier werden die Blöcke platziert. Die Linien zwischen den Slots dienen danach als Verknüpfung der unterschiedlichen Blöcke.
- DELETE-Button: Mit diesem Button kann der aktuelle Levelfortschritt zurückgesetzt und es kann von vorne begonnen werden, wenn man einen Fehler gemacht hat.
- START-Button: Sind alle Blöcke platziert, die man platzieren wollte, kann dieser Button gedrückt werden, um das Programm zu starten. Nun wird geprüft, ob das Programm funktionsfähig ist und die Aufgabe somit erfolgreich bearbeitet wurde.
- HELP-Button: Mit diesem Button lässt sich ein Pop-Up öffnen, das Hilfestellungen zur jeweiligen Aufgabe bietet.
- BACK-Button: Mit diesem Button kommt man zurück zur vorherigen Ansicht (Menüs).

C.2.4 Blöcke:

- START-Block: Dieser Block ist statisch und symbolisiert den Programmstart.
- ZIEL-Block: Dieser Block ist ebenfalls statisch und symbolisiert das Ende des Programms.

- WHILE-Block: Dieser Block erhöht eine Variable (hier: x) so lange, bis der im Block angegebene Wert erreicht wurde und leitet das Signal dann zum nächsten Block weiter.
- IF-Block: Dieser Block prüft, ob eine Variable (hier: x) einen gegebenen Wert hat oder nicht, je nachdem wird entweder der rote (false) oder der grüne (true) Ausgang verwendet.
- END-Block: Dieser Block wird symbolisch für den nicht verwendeten Ausgang eines IF-Blocks verwendet (hier meist “false”).

C.2.5 Zusammenfassung:

Die verwendeten Blöcke symbolisieren unterschiedliche Strukturen und Funktionen, die man in fast jeder Programmiersprache wiederfindet. Ziel dieser Anwendung ist es, zu verstehen, wie diese Funktionen und Strukturen arbeiten, da diese universell auf alle Programmiersprachen anwendbar sind. So lassen sich mit CodeClash Programme auch ohne eine komplexere Programmiersprache schreiben, da für die Funktionen symbolisch einzelne Blöcke verwendet werden.

D Anhang - Farbzauber und Funktionen

D.1 Anhang - SchülerPDF

Farbzauber und Funktionen -

eine experimentelle Einführung in die Programmierung

1 Lernziele

In dieser Übung sollst du mittels einfacher, kleiner Schritte ein Einblick in die Programmierung, vor allem in den Aufruf und die Verwendung von Methoden, bekommen. Dabei solltest du aktiv mit kleinen Programmen experimentieren. Durch Ausprobieren und mithilfe deiner Kommilitonen^[^1] und des Dozenten^[^2] sollte am Ende ein kleines Bild dabei rauskommen. Mit der Bearbeitung der Aufgaben sollen folgenden Fähigkeiten erlangt werden:

- Verständnis für den Begriff 'Methoden' im Kontext der Programmierung sind und wie diese verwendet werden
- Erzeugung, Berechnung und Darstellung von Farben auf dem Bildschirm
- Programmierung von einfachen Berechnungen auf Zahlen und diese in einer Schleife automatisch ausführen
- Eine Einleitung in weitere grundlegende Konzepte der Programmierung und ggf. Verwendung und Vertiefung dieser.

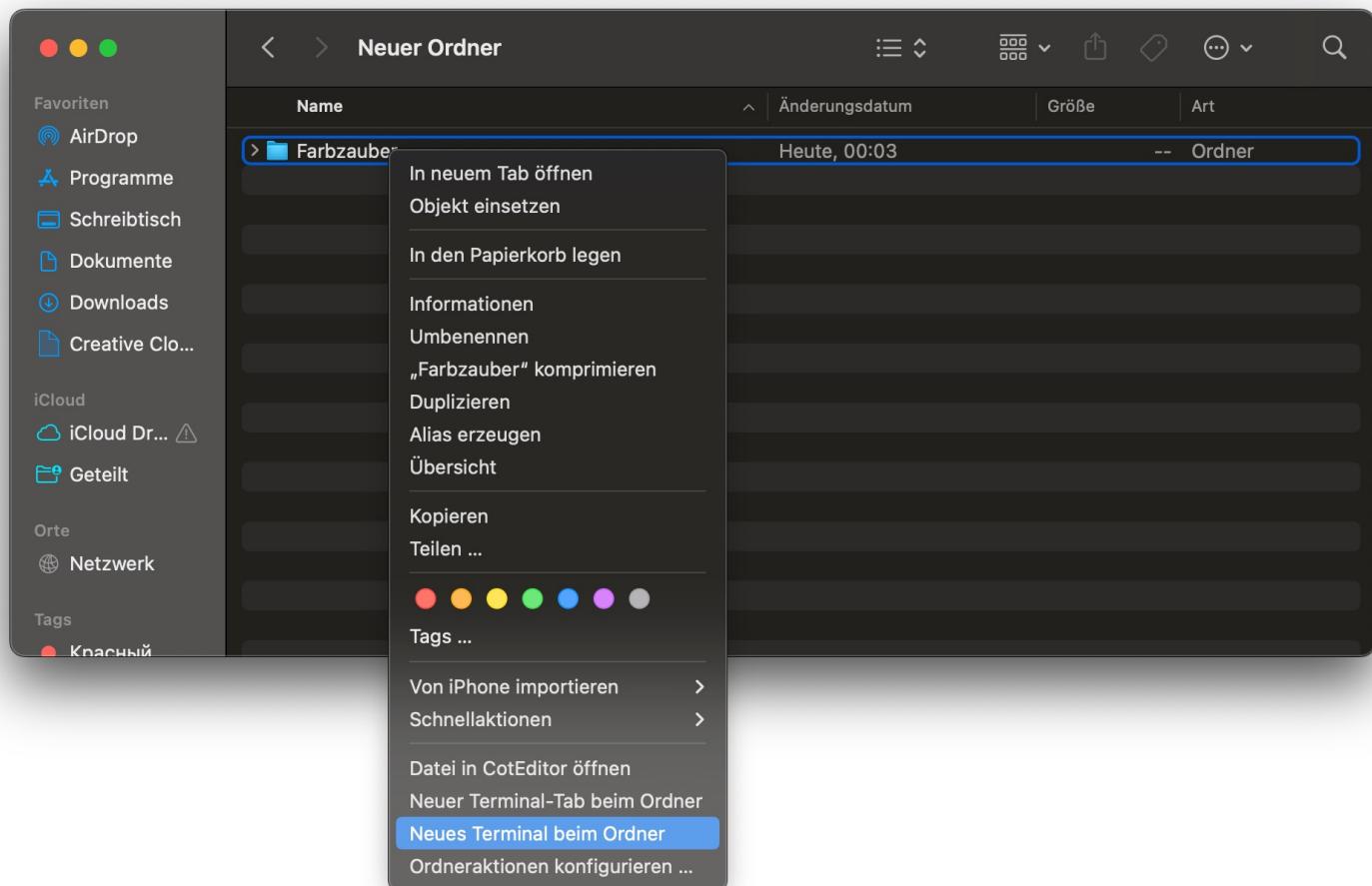
Dabei dieses Ziel zu erreichen kann helfen:

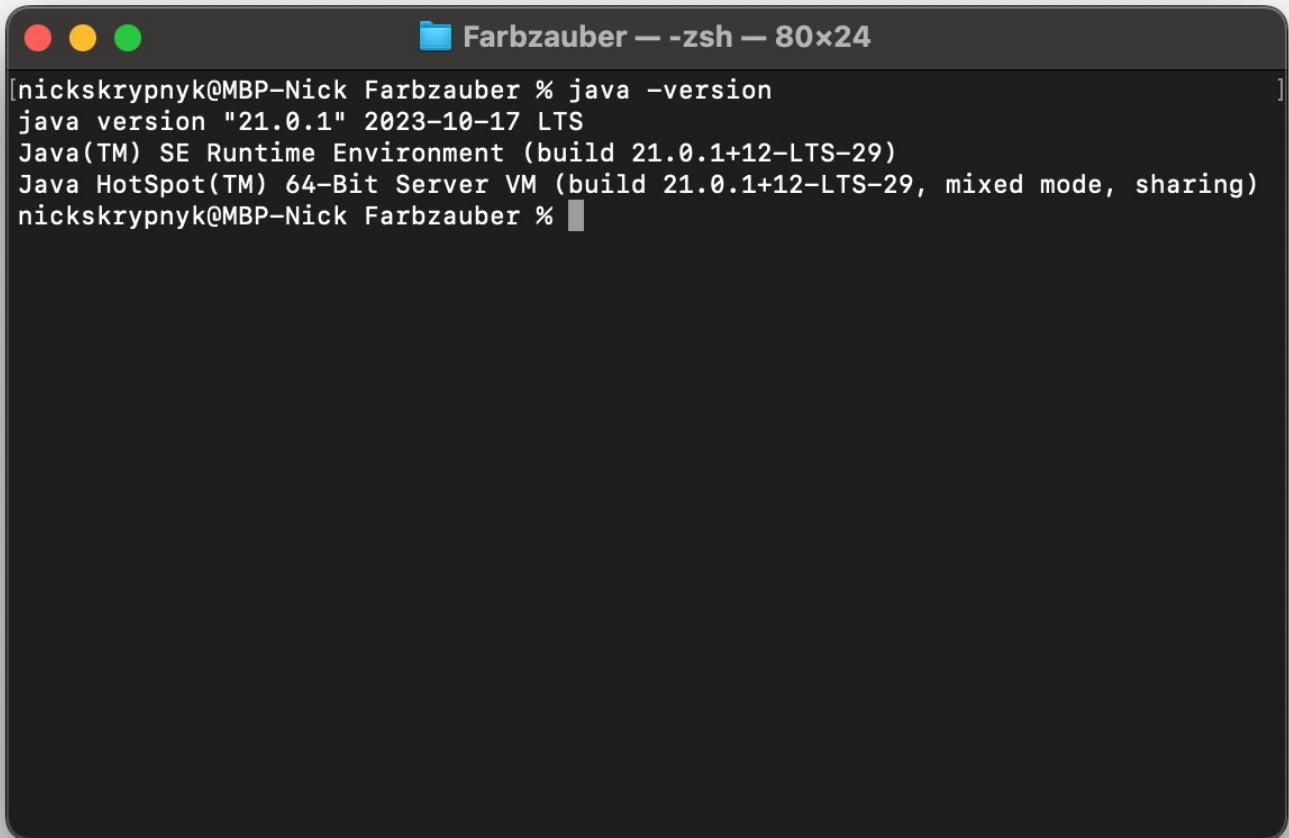
- Immer langsam, je mehr man versteht, umso weniger muss auswendig gelernt werden
- Darüber reden, laut am besten mal mit dem Sitznachbar
- Notizen machen am besten mit einem Stift, nicht nur die Übungen ansehen
- Viel trinken, denn das Gehirn braucht Wasser. (Aber bitte nicht hier am Platz)

Hilft auch im normalen Unterricht und bei den Hausaufgaben

2 Vorbereitung

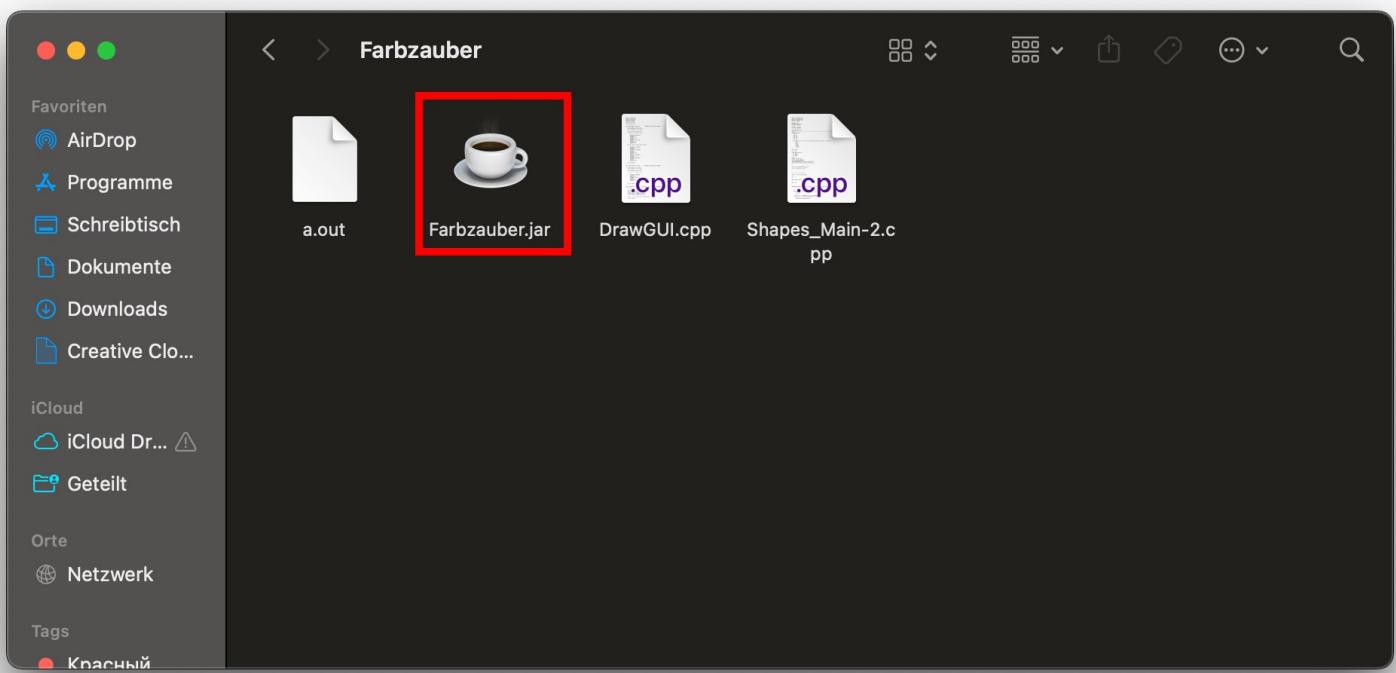
Auf dem Desktop befindet sich ein Ordner namens Programmiererlebnisse, zunächst klickst du einmal mit der rechten Maustaste darauf und wählst *Neues Terminal beim Ordner aus*



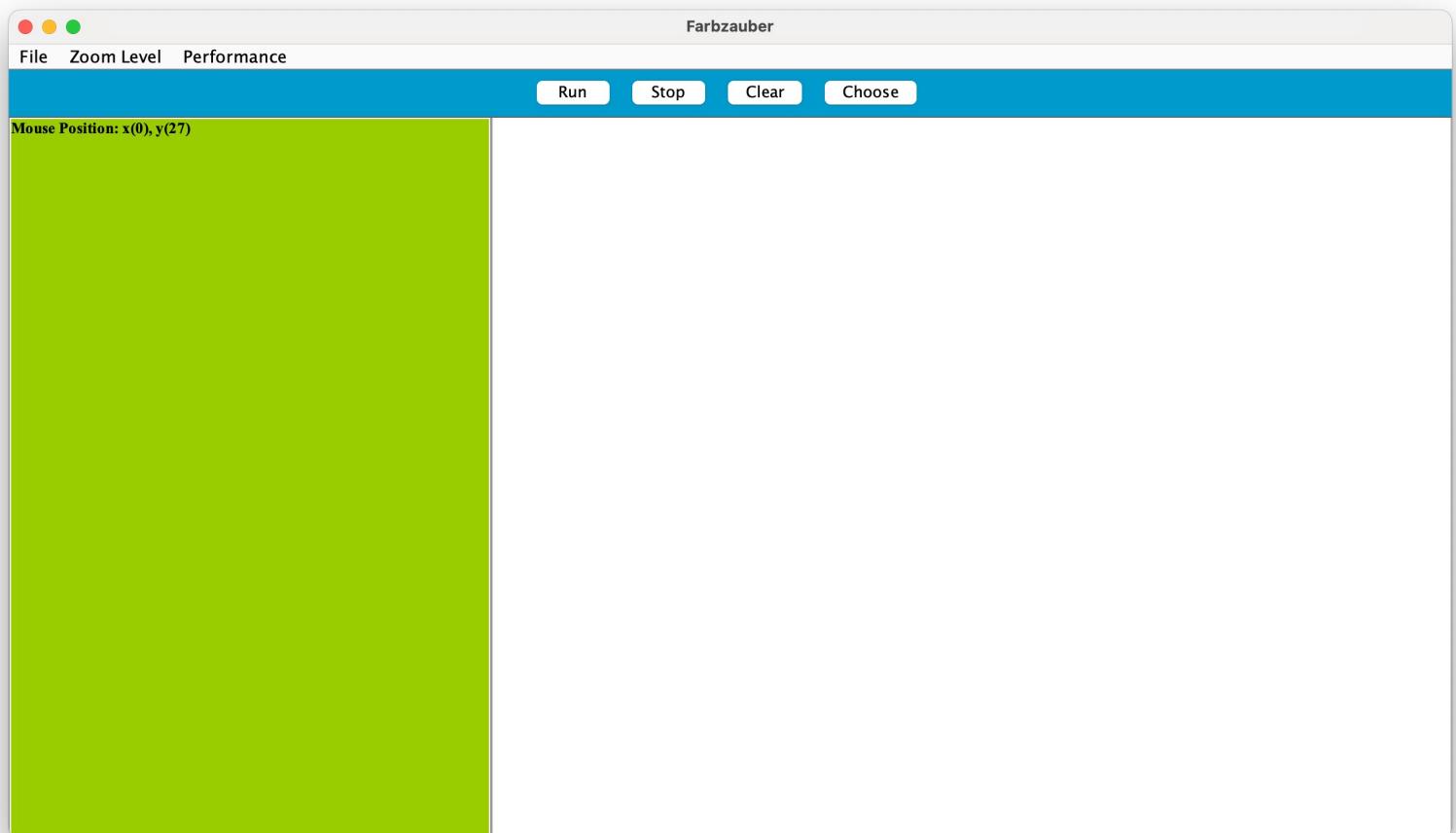
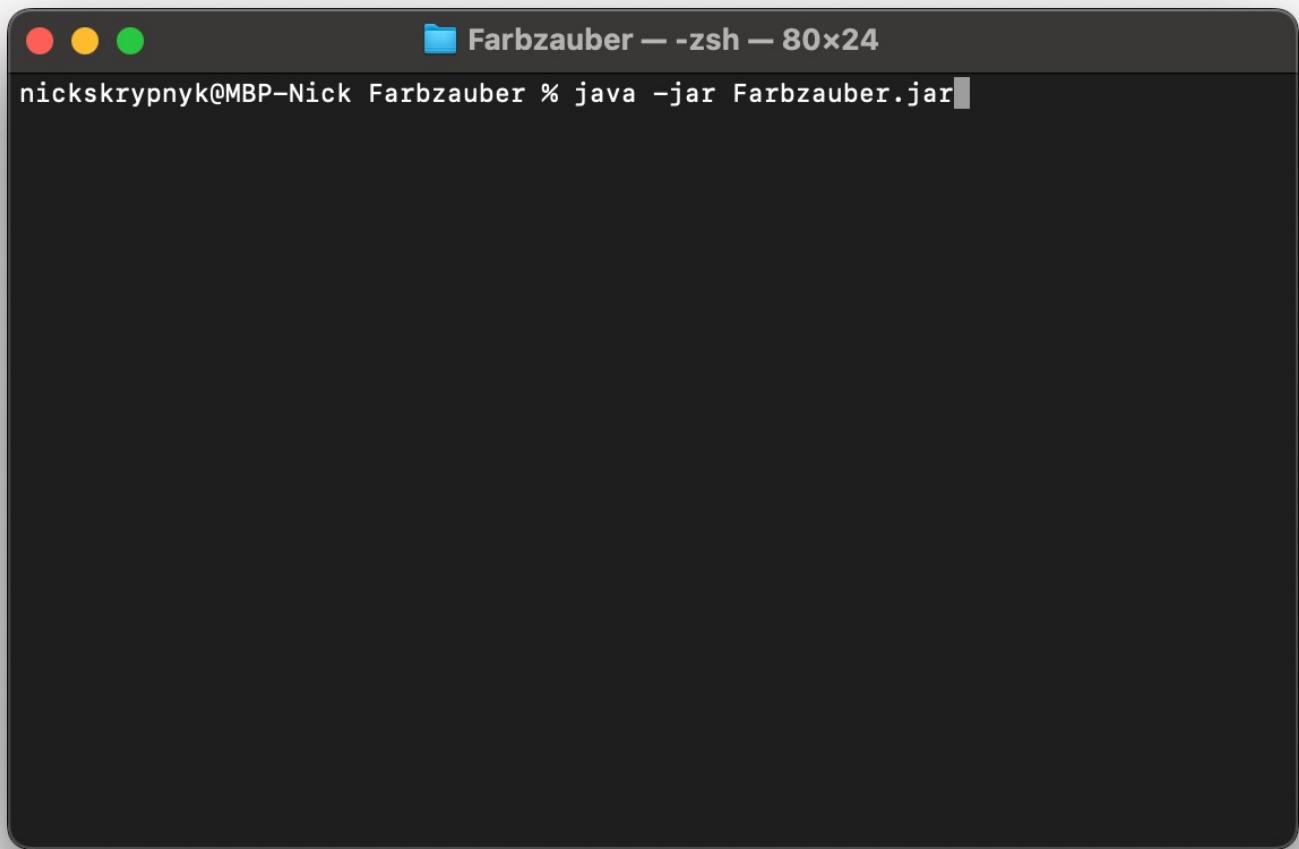


```
[nickskrypnyk@MBP-Nick Farbzauber % java -version
java version "21.0.1" 2023-10-17 LTS
Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.1+12-LTS-29, mixed mode, sharing)
nickskrypnyk@MBP-Nick Farbzauber % ]
```

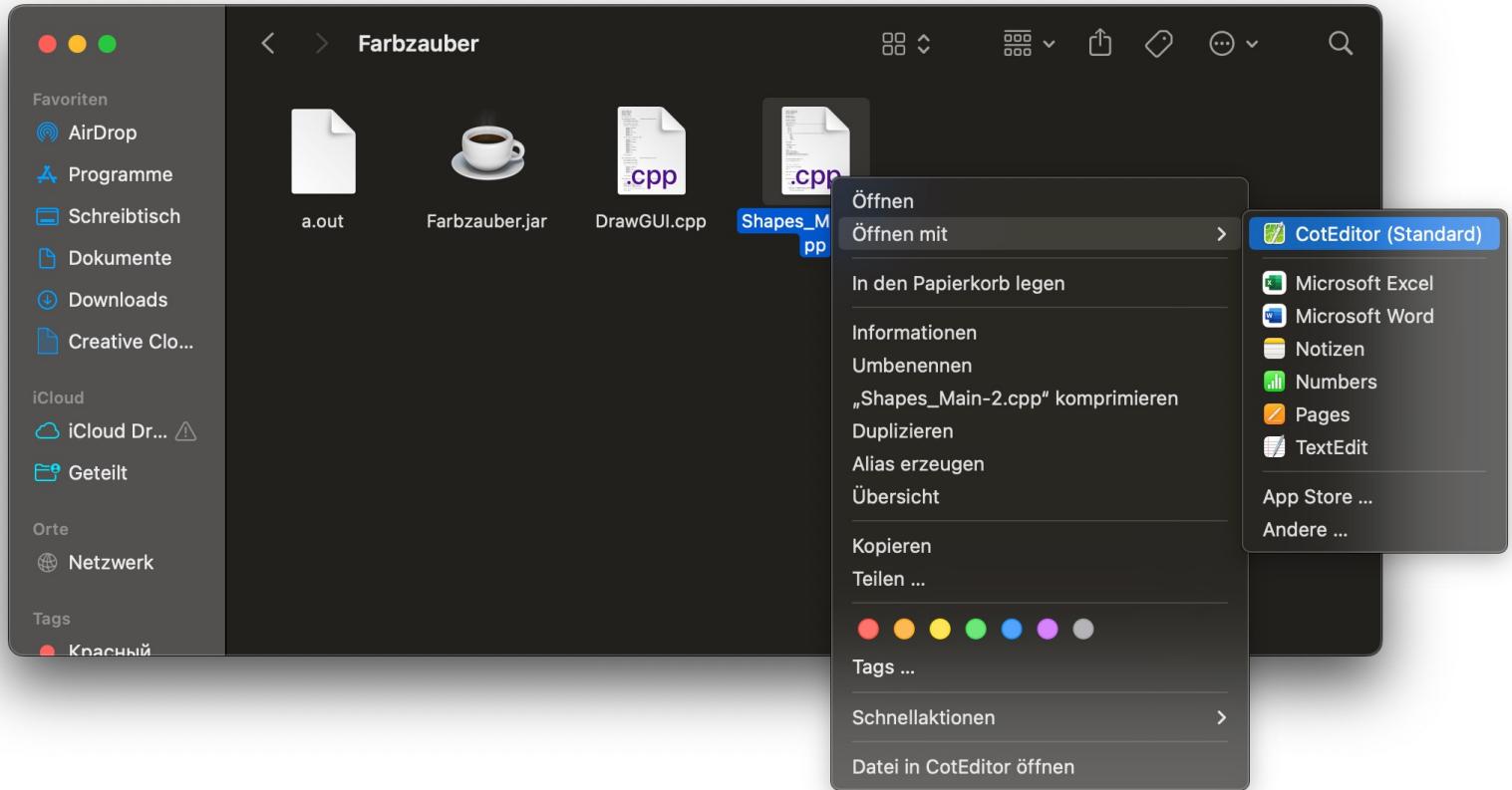
Danach öffne den Ordner und dort sollte ein Programm namens *Farbzauber.jar*, die aussieht wie eine heiße Tasse Kaffee, zu finden sein.



Wenn du diese öffnest, erscheint unsere Leinwand. Sollte dem nicht der Fall sein und du bekommst stattdessen eine Fehlermeldung, dann starte das Programm übers Terminal mit dem Kommando `Farbzauber % java -jar Farbzauber.jar &`



und zum Schluss öffnet man noch die *Shapes_Main.cpp* im Editor in dem wir einen Rechtsklick auf die Datei machen und dann mit der Maus über "Öffnen mit" hovern und dann in der neuen Auswahl "CotEditor" auswählen.



Hier scrollen wir einmal nach unten bis wir bei `int main() {` an gelangen, wo wir später auch unsere gesamte Arbeit verrichten. Du kannst im Übrigen alles in diesem Dokument was in solchen Code-blöcken steht:

Hallo Welt

ganze einfach mit "⌘ + C" und "⌘ + V" kopieren und nun kann man im Terminal folgendes eingeben:

```
clang Shapes-Main.cpp -o a
```

womit im selben Ordner wie die `Shapes_Main.cpp` eine Datei mit dem Namen `a.out` erscheinen sollte.

welche sich mit dem "Choose" Button unserer Leinwand öffnen lässt und uns unser erstes Bild zeichnet.

3 Variablen

Wie eine Box mit Namen. Man kann den Inhalt ersetzen, bearbeiten oder an verschiedenen Stellen darauf zugreifen. Nützlich, wenn man an verschiedenen Stellen denselben Wert benötigt oder verändern möchte.

Erstellen z.B. für ganze Zahlen: `Int zahl_1; Int zahl_2; Int ergebniss;`

Zahlen darin abspeichern, verändern und an mehreren Stellen verwenden: `zahl_1 = 1; zahl_2 = 3;`

```
zahl_1 = zahl_1 + 1;
zahl_1 += 1;
ergebniss = zahl_1 * zahl_2;
```

Dabei ist zu beachten, dass diese Variablen nur eine bestimmte Lebensdauer haben, sprich sie existieren nur im Rahmen von der zuletzt geöffneten geschweiften Klammer "{" bis hin wo diese wieder geschlossen wird "}"

Weitere Typen: Float und Double für Kommazahlen. Boolean: Ist in diesem Fall entweder 0 für FALSE oder ein anderer wert für TRUE. Gut verwendbar für If-Abfragen. Dabei ist zu beachten, dass man für einen "ist gleich" Vergleich "==" verwendet.

```
bool nee = 0;
bool jap = 1;
```

```
if(nee == 0){
    zahl_1 -= 1;
}
```

Kurz `if(jap) zahl_1 += 1;`

Ist `zahl_1` jetzt um eins größer?

Char: Um einen einzelnen character also Buchstaben, Zeichen, Ziffern zu Speichern. String: Ist eine verkettung von Char. Das heißt der String "Moin" besteht aus den Char 'M', 'o', 'i', 'n' in der Reihenfolge.

Arrays: Gut, wenn etwas öfter in gleicher Reihenfolge benötigt wird. Nacher werden noch Schleifen genannt z.B. die for-Schleife mit denen Arrays leicht zu verwenden sind.

```
int triangle_position_x[4];
triangle_position_x[0] = 200;
triangle_position_x[1] = 300;
triangle_position_x[2] = 340;
triangle_position_x[3] = 300;
```

Beim aufrufen nutzt man den Arraynamen(`triangle_position_x`) mit passendem Index([1]). dabei ist darauf zu achten das der Index bei null anfängt und nicht überschritten wird.

4 Farben auf dem Bildschirm

Farben

Werden RGB also (Red, Green, Blue) dargestellt. Dabei handelt es sich um Additive Farbmischung die beim Mischen von Licht angewandt wird. In unserem Fall mit drei Werten die von 0 bis 255 gehen. Sind alle drei Werte gleich, hat man Graustufen. Dabei ist (255, 255, 255) Weiß und (0, 0, 0) Schwarz. Ein paar Farbbeispiele wären: Rot(255, 0, 0), Grün(0, 255, 0), Blau(0, 0, 255), Gelb(255, 255, 0), Orange(255, 127, 0), Magenta(255, 0, 255)

Hier ein Link zum Ausprobieren: https://informatik.schule.de/rgb/RGB_farbmaischer.html (https://informatik.schule.de/rgb/RGB_farbmaischer.html)

5 Funktionen

Ähnlich wie bei Variablen sind Funktionen Programmabschnitte, die über ihren Namen mehrmals, an verschiedenen Stellen, aufrufbar sind. Häufig werden Parameter, meist Werte, mitgegeben bzw. zurückgegeben.

Erstellen z.B. um zwei übergebene ganze Zahlen (Variablen nach dem Funktionsnamen) zu addieren und zurückzugeben (inhalt der Klammer nach return):

```
int addiere(int summand1, int summand2) {
    int summe = summand1 + summand2;
    return (summe);
}
```

Aufrufen der Funktion z.B. um ein Ergebnis der Zahlen 3 und 7 zu erhalten:

```
int summe = addiere(3, 7);
```

Wir stellen euch schon die Funktion zum Setzen eines Pixels zur Verfügung. Pixel: (position_x, position_y, rot, grün, blau) setPixel(2, 2, 50, 50, 50);

Als weiteren Teil der Funktionen gibt es noch Konstruktoren. Diese funktionieren wie ein Bauplan für eigene Variablen bzw. Objekte und können mehrere Variablen und Funktionen enthalten.

Aufrufen eines Konstruktor z.B. für einen Kreis

```
Circle* circle = new Circle(0, 0, 200, 0, 0, 0, 5);
```

In diesem Fall wurde ein Kreis(Circle*) der den Namen(circle) hat mit der Funktion(Circle) und den Parametern((0,0,200,0,0,0,5)) erstellt.

In diesem Fall haben wir euch einige bereitgestellt.

Linie: (anfang_x, anfang_y, ende_x, ende_y, rot, grün, blau, linienbreite)

```
Line* line = new Line(45, 55, 25, 15, 100, 0, 255, 5);
```

Dreieck: (ecke_1_x, ecke_1_y, ecke_2_x, ecke_2_y, ecke_3_x, ecke_3_y, rot, grün, blau, linienbreite)

```
Triangle* triangle = new Triangle(11, 11, 22, 22, 11, 22, 100, 100, 10, 5);
```

Kreis: (ecke_oen_links_x, ecke_oen_links_y, radius, rot, grün, blau, liniendicke)

```
Circle* circle = new Circle(0, 0, 200, 0, 0, 0, 5);
```

Rechteck: (ecke_oen_links_x, ecke_oen_links_y, breite, Höhe, rot, grün, blau, linienbreite)

```
Rectangle* rectangle = new Rectangle(0, 20, 80, 30, 255, 255, 255, 5);
```

Text: (ecke_oen_links_x, ecke_oen_links_y, "Hier der Text", "Schriftart", schriftgröße, 0_keines/1_dick/2_kursiv/3_beides, rot, grün, blau)

```
StringText* stringText = new StringText(25, 10, "Moin", "Comic Sans", 12, 1, 50, 50, 255);
```

6 Schleifen

Du kannst auch einmal versuchen den Kreis oder die Linie mehrmals aufzurufen, wobei du vor allem darauf achten musst jeden neuen Kreis einen eigenen Namen zu geben zum Beispiel:

```
//Kreis
Circle* gruen = new Circle(400,250,50,0,255,0,5);
gruen->draw();
Circle* gruen2 = new Circle(400,350,50,0,255,0,5);
gruen2->fill();
Circle* gruen3 = new Circle(400,450,50,0,255,0,5);
gruen3->draw();
```

Erkennst du vielleicht schon das Problem? Was ist, wenn du 10 Kreise oder vielleicht sogar 100 neue Kreise zeichnen wolltest?

Für solche Fälle gibt es ein Konzept in der Programmierung namens Schleifen, es gibt vor allem Drei Varianten:

Die while-Schleife, welche stets erst eine gegebene Bedingung überprüft bevor sie ausführt:

```
while(Bedingung = wahr){
    mach etwas
}
```

Die Do-While-Schleife, welche im Gegensatz erst einmal ausführt bevor sie die gegebene Bedingung ausgeführt wird:

```
do{
    mach etwas
}while(Bedingung = wahr)
```

Und die for-Schleife welche auch die zählende Schleife genannt wird, da sie oft genutzt wird um eine Aktion eine bestimmte Anzahl von Malen auszuführen.

```
for(Startwert;Bedingung;Schrittweite){
    mach etwas
}
```

Für unsere Zwecke eignet sich vor allem die for-Schleife, weil mit dieser lässt sich der Code von oben so vereinfachen

```
for(int i = 250; i <= 450; i = i + 100){
    Circle* gruen = new Circle(400,i,50,0,255,0,5);
    gruen->draw();
    delete gruen;
}
```

Außerdem müssen wir nun auch darauf achten unser erzeugtes Objekt auch wieder zu löschen damit wir es noch einmal neu an anderer Stelle neu zu zeichnen.

7 If-Abfragen

Aber vielleicht fällt dir ein Unterschied zwischen der vorherigen Ausgabe und jetzt auf, der mittlere Kreis ist nicht mehr ausgefüllt! Da müssen wir wohl eine bedingte Ausnahme festlegen. Hier kommen nun sogenannte "if-Abfragen" ins Spiel.

```
if(Bedingung == wahr){  
    mach etwas  
}  
else{  
    mach etwas Anderes  
}
```

Man kann auch mehrere verschiedene Abfragen aufeinander stapeln welche jeweils nur ausgeführt werden, wenn die vorherigen Abfragen unerfüllt blieben.

```
if(Bedingung == wahr){  
    mach etwas  
}  
else if( Andere Bedingung == wahr){  
    mach etwas Anderes  
}  
else{  
    mach etwas ganz Anderes  
}
```

Dies erlaubt uns verschiedene Aktionen basierend auf einer Bedingung auszuführen zum Beispiel unseren mittleren Kreis zu füllen.

```
for(int i = 250; i <= 450; i = i + 100){  
    Circle* gruen = new Circle(400,i,50,0,255,0,5);  
    if( i == 350){  
        gruen->fill();  
    }else {  
        gruen->draw();  
    }  
    delete gruen;  
}
```

8 Getting Started

Wenn man nun in der *Shapes_Main.cpp* im Editor nach unten scrollt bis `int main(){` findet man folgende Aufrufe und Methoden:

```
//Rechteck  
Rectangle* blau = new Rectangle(100,200,50,50,0,0,255,10);  
blau->fill();  
  
//Dreick  
Triangle* weiss = new Triangle(750,250,500,150,600,250,255,255,255,10);  
weiss->draw();  
  
//Text  
StringText* stringText = new StringText(250, 0, "Hallo Welt", "Comic Sans", 24, 1,100,0,255);  
stringText->draw();  
  
//Linie  
Line* rot = new Line(250, 250, 250, 150, 255, 0, 0, 5);  
rot->draw();  
  
//Kreis  
Circle* gruen = new Circle(400,250,50,0,255,0,2);  
gruen->draw();  
  
//Pixel  
setPixel(325,250,0,0,0);
```

/* Zeigt einen Kommentar an, dieser wird von einem Programm nicht beachtet, sondern dient lediglich nur dir als Programmierer

"`Rectangle* blau = new Rectangle`" ruft einen sogenannten Konstruktor auf welcher für uns ein neues Rechteck mit dem Namen "blau" erstellt.

Der interessante Teil hierbei sind die Zahlen die wir als sogenannte "Parameter" dabei übergeben. Wenn man diese Verändert, verändert sich das Rechteck. Probier es einfach mal aus und verändere einige der Parameter und schaue was passiert aber Achtung, einige der Werte können nicht höher als 255 gesetzt werden, welche das sind und warum klären wir später.

`blau->fill();` zeichnet dann tatsächlich das Rechteck bzw. `blau->draw();` würde dann nur den Umriss zeichnen wie du gut an dem Dreieck oder Kreis sehen kannst

Speichere die Änderungen, keine Sorge du kannst alle Änderungen mit dem Tastenkürzel "⌘ + Z" jederzeit rückgängig machen so lange du den Editor nicht schließt. Übersetze noch einmal mit dem Terminal mit `clang Shapes-Main.cpp -o a` und dann lass dir mit dem "Choose" Button auf der Leinwand die Änderungen zeichnen. Sollten dir dabei Fehlermeldungen anzeigen dann frag doch einmal den Dozenten.

9 Konventionen

Zum Schluss solltest du noch ein paar Konventionen[^5] und sogenannte "good practices" also gute Gewohnheiten beim Programmieren mitnehmen von denen du auch so einige hier schon gesehen hast

Variablennamen sollten eindeutig sein und werden meist kleingeschrieben, wobei es mehrere Möglichkeiten gibt, hier nur mal zwei.

-> erhöht die Lesbarkeit und hilft beim Verständnis

`pixelPositionY = 125;`

oder

`pixel_position_y = 125;`

Code zwischen {} sollte mit der tab Taste stets (weiter) eingerückt werden
Zudem werden geschweifte Klammern "{" bzw. "}" direkt hinter die Funktion geschrieben bzw. in dieselbe Reihe beim Schließen
-> erhöht die Lesbarkeit deutlich

```
if(x == true){  
    if(y == true){  
        do;  
    }  
}
```

Vergesse nicht Kommentare zu schreiben und achte darauf die Umlaute wie ä ö ü vermieden werden sollten

-> Hilft deutlich beim Verständnis und Umlaute werden nicht von allen Texteditoren oder Entwicklungsumgebungen unterstützt

/Hier koennte ihr Kommentar stehen

Somit solltest du eigentlich gewappnet[^6] sein das, wahrscheinlich von dem Dozenten vorgegeben Bild, gut zu zeichnen oder deiner kreativität freien lauf zu lassen.

Viel Glück und Erfolg!

10 Glossar

[^1]: Jemand mit dir zur selben Zeit zur selben Schule geht. [^2]: Lehrer an einer Hochschule. [^3]: Datei Explorer nur für Mac. [^4]:⌘ wird uch CMD oder Command Taste genannt.
[^5]: Regeln oder Vereinbarungen für das (soziale) Verhalten. [^6]: Gut ausgerüstet/ausgestattet, um schwierige Situationen zu überstehen

D.2 Anhang - LehrerPDF

1 Installationsanleitung

Um eine reibungslose Inbetriebnahme der Anwendung zu gewährleisten ist vorher sicherzustellen, dass eine Entwicklungsumgebung für C++ eingerichtet und Java JDK 21 oder neuer installiert ist

1.1 C++ Entwicklungsumgebung

1.1.1 Windows

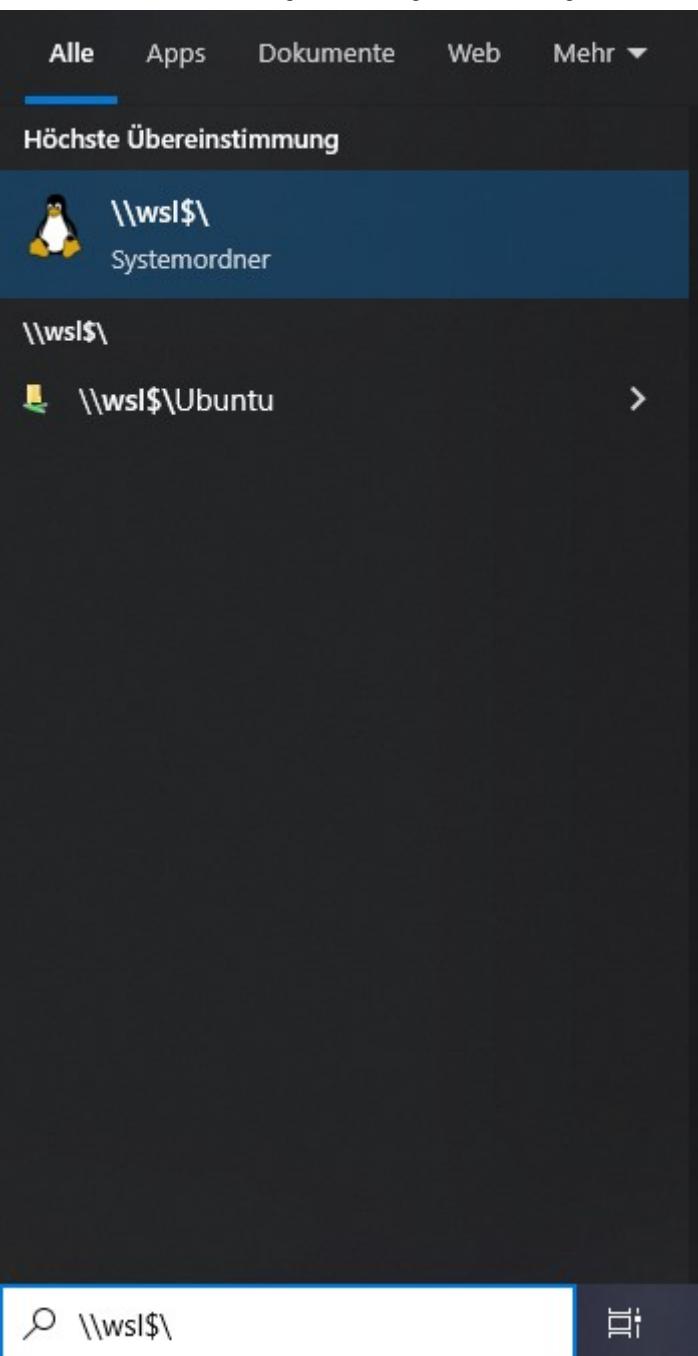
Wenn sie über Windows 10, Version 2004 und höher (Build 19041 und höher), oder Windows 11 verfügen öffnen sie zunächst das Terminal und geben sie

```
wsl --install
```

ein um das "Windows Subsystem for Linux (WSL)" zu installieren. Bei älteren Windows Installationen folgen Sie bitte [dieser Anleitung](https://learn.microsoft.com/de-de/windows/wsl/install-manual) (<https://learn.microsoft.com/de-de/windows/wsl/install-manual>)

Hierach ist ein Systemneustart erforderlich. Folgen Sie anschließend den Anweisungen auf dem Bildschirm.

Ab hier können sie der Anleitung für Linux folgen mit dem einzigen Unterschied, dass sie über die Suche mit \\wsl\\$\\ den Dateiordner öffnen müssen



1.1.2 Linux

Öffnen sie das Terminal und führen sie folgende Befehle nacheinander aus

```
$ sudo apt-get update  
$ sudo apt-get upgrade  
$ sudo apt-get install build-essential  
$ sudo apt-get install gcc-multilib
```

Nun sollten Sie eine funktionsfähige gcc-Installation zur Verfügung haben. Mit

```
$ gcc -v
```

können Sie Ihre Installation testen.

1.1.3 MacOS

Öffnen sie das Terminal und geben sie dort

`clang --version`

ein und folgen sie den weiteren Anweisungen auf dem Bildschirm

1.2 Java JDK

1.2.1 Linux & Windows

Öffnen sie das Terminal und geben sie folgenden Befehl ein

`sudo apt install default-jdk.`

Nun sollten Sie eine funktionsfähige Java-Installation zur Verfügung haben. Mit

`java -version`

können Sie Ihre Installation testen.

1.2 Java JDK

Wählen sie unter [diesen Link](https://www.oracle.com/de/java/technologies/downloads/#jdk21-mac) (<https://www.oracle.com/de/java/technologies/downloads/#jdk21-mac>), oder eine andere Distribution, die passende Version für ihr System und folgen sie den weiteren Anweisungen auf dem Bildschirm.

2 Musterlösungen

2.1 Weihnachtsbild

```

int main() {

    //Rectangle
    Rectangle* schnee = new Rectangle(0,399,799,200,255,255,255,10);
    Rectangle* schnee2 = new Rectangle(0,399,799,200,0,0,0,1);
    schnee->fill();
    schnee2->draw();
    delete schnee;
    delete schnee2;

    //Rectangle::Rectangle(int x, int y, int width,int height,int red,int green,int blue, int lineWidth){

        Rectangle* stump = new Rectangle(125,250,50,200,139,69,19,10);
        Rectangle* stump2 = new Rectangle(125,250,50,200,0,0,0,1);
        stump->fill();
        stump2->draw();

        int x1 = 100; int y = 150; int x2 = 200; int y3 = 100;

        for (int i = 1; i <= 4; i++){

            Triangle* t = new Triangle(x1, y, x2, y, 150, y3, 0, 100, 0, 5);
            Triangle* t2 = new Triangle(x1, y, x2, y, 150, y3, 0, 0, 0, 1);
            t->fill();
            t2->draw();
            delete t;
            delete t2;

            x1 = x1 - 25;
            y = y + 50;
            x2 = x2 + 25;

            y3 = y3 + 30;
        }

        //StringText
        StringText* stringText = new StringText(250, 0, "Frohe Weihnachten", "Comic Sans", 24, 1,100,0,255);
        stringText->draw();

        //Circle
        int r = 75; int circleX = 550; int circleY =375;
        int values[] = {375, 300, 250};
        for (int i = 1; i <= 3; i++){

            Circle* c = new Circle(circleX,circleY,r,255,255,255,5);
            Circle* c2 = new Circle(circleX,circleY,r,0,0,0,1);
            c->fill();
            c2->draw();
            delete c;
            delete c2;

            r = r - 25;
            circleX = circleX + 25;
            circleY = values[i];
        }

        int knopfY = 425;
        for (int i = 1; i <= 3; i++){
            Circle* c = new Circle(615,knopfY,10,255,94,5,5);
            Circle* c2 = new Circle(615,knopfY,10,0,0,0,1);
            c->fill();
            c2->draw();
            delete c;
            delete c2;
            knopfY = knopfY -50;
        }

        int AugeX = 610;
        for (int i = 1; i <= 2; i++){
            Circle* c = new Circle(AugeX,265,5,0,0,0,5);
        }
    }
}

```

```
c->fill();
delete c;
AugeX = AugeX + 20;
}

//Rectangle::Rectangle(int x, int y, int width,int height,int red,int green,int blue, int lineWidth)
Rectangle* hat1 = new Rectangle(575,250,100,10,0,0,0,10);
hat1->fill();

Rectangle* hat2 = new Rectangle(600,200,50,50,0,0,0,10);
hat2->fill();

//Triangle      (100,500,200,300,300,500, new Color(102,0,153), 10)
Triangle* nose = new Triangle(625, 280, 625, 290, 600, 285, 255, 165, 0, 5);
nose->fill();

Line* arm_links = new Line(575, 340, 530, 300, 0, 0, 0, 5);
arm_links->draw();
Line* arm_rechts = new Line(670, 340, 720, 300, 0, 0, 0, 5);
arm_rechts->draw();

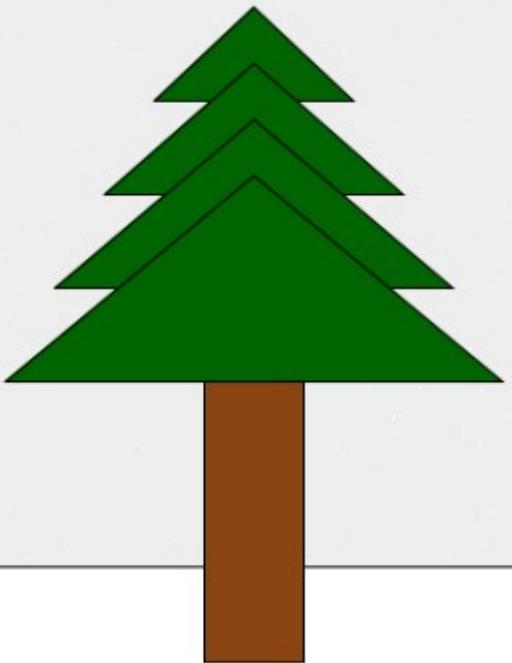
//pixel

std::random_device rd;
std::mt19937 gen(rd());

std::uniform_int_distribution<int> distribution1(0,799);
std::uniform_int_distribution<int> distribution2(0,599);

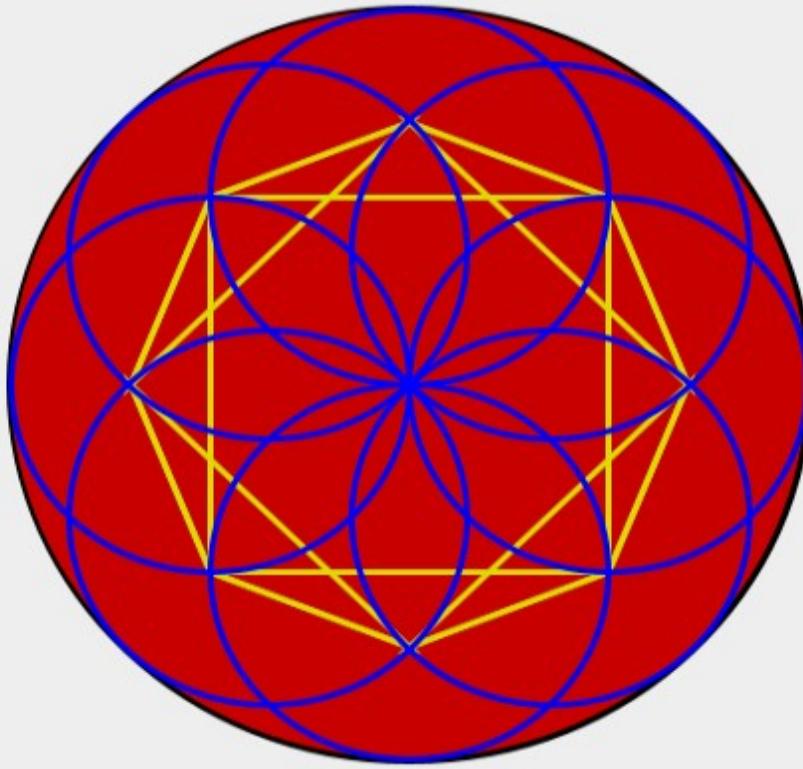
for(int i= 0; i<1000 ; i++) {
    Pixel* p = new Pixel(distribution1(gen),distribution2(gen),255,255,255);
    p->draw();
    delete p;
}
return 0;
}
```

Frohe Weihnachten



2.2 Mandala

```
int main() {  
  
    //Circle  
  
    Circle* c0 = new Circle(150,100,200,0,0,0,5);  
    c0->draw();  
  
    Circle* c1 = new Circle(150,100,200,200,0,0,5);  
    c1->fill();  
  
    //Triangle  
  
    int triangle_position_x[10];  
    triangle_position_x[0] = 350;  
    triangle_position_x[1] = 450;  
    triangle_position_x[2] = 490;  
    triangle_position_x[3] = 450;  
    triangle_position_x[4] = 350;  
    triangle_position_x[5] = 250;  
    triangle_position_x[6] = 210;  
    triangle_position_x[7] = 250;  
    triangle_position_x[8] = 350;  
    triangle_position_x[9] = 450;  
  
    int triangle_position_y[10];  
    triangle_position_y[0] = 160;  
    triangle_position_y[1] = 200;  
    triangle_position_y[2] = 300;  
    triangle_position_y[3] = 400;  
    triangle_position_y[4] = 440;  
    triangle_position_y[5] = 400;  
    triangle_position_y[6] = 300;  
    triangle_position_y[7] = 200;  
    triangle_position_y[8] = 160;  
    triangle_position_y[9] = 200;  
  
    for(int i = 0; i<8; i++){  
        Triangle* t1 = new Triangle(triangle_position_x[i], triangle_position_y[i], triangle_position_x[i+1], triangle_position_y[i+1],  
        triangle_position_x[i+2], triangle_position_y[i+2], 220, 220, 0, 3);  
        t1->draw();  
    }  
  
    //Circle  
  
    int circle_position_x;  
    int circle_position_y;  
  
    Circle* c001 = new Circle(250,100,100,0,0,255,3);  
    c001->draw();  
    Circle* c002 = new Circle(150,200,100,0,0,255,3);  
    c002->draw();  
    Circle* c003 = new Circle(350,200,100,0,0,255,3);  
    c003->draw();  
    Circle* c004 = new Circle(250,300,100,0,0,255,3);  
    c004->draw();  
  
    circle_position_x = 179;  
    for(int i = 1; i <= 2; i++){  
        circle_position_y = 129;  
        for(int j = 1; j <= 2; j++){  
            Circle* c06 = new Circle(circle_position_x,circle_position_y,100,0,0,255,3);  
            c06->draw();  
            circle_position_y += 142;  
        }  
        circle_position_x = circle_position_x + 142;  
    }  
  
    return 0;  
}
```



2.3 RGB

```
int main() {
    //Circle(int x, int y, int radius,int red,int green,int blue, int lineWidth)
    Circle* c01 = new Circle(70,50,50,0,0,0,3);
    c01->draw();
    delete c01;
    Circle* c02 = new Circle(270,50,50,0,0,0,3);
    c02->draw();
    delete c02;
    Circle* c03 = new Circle(470,50,50,0,0,0,3);
    c03->draw();
    delete c03;
    Circle* c04 = new Circle(670,50,50,0,0,0,3);
    c04->draw();
    delete c04;

    Circle* c11 = new Circle(70,50,50,0,255,255,3);
    c11->fill();
    delete c11;
    Circle* c12 = new Circle(270,50,50,0,127,255,3);
    c12->fill();
    delete c12;
    Circle* c13 = new Circle(470,50,50,0,0,255,3);
    c13->fill();
    delete c13;
    Circle* c14 = new Circle(670,50,50,127,0,255,3);
    c14->fill();
    delete c14;
```

```

//Rectangle(int x, int y, int width,int height,int red,int green,int blue, int lineWidth)
Rectangle* r01 = new Rectangle(70,250,100,100,0,0,0,3);
r01->draw();
delete r01;
Rectangle* r02 = new Rectangle(270,250,100,100,0,0,0,3);
r02->draw();
delete r02;
Rectangle* r03 = new Rectangle(470,250,100,100,0,0,0,3);
r03->draw();
delete r03;
Rectangle* r04 = new Rectangle(670,250,100,100,0,0,0,3);
r04->draw();
delete r04;

Rectangle* r11 = new Rectangle(70,250,100,100,255,0,255,3);
r11->fill();
delete r11;
Rectangle* r12 = new Rectangle(270,250,100,100,255,0,127,3);
r12->fill();
delete r12;
Rectangle* r13 = new Rectangle(470,250,100,100,255,0,0,3);
r13->fill();
delete r13;
Rectangle* r14 = new Rectangle(670,250,100,100,255,127,0,3);
r14->fill();
delete r14;
//Triangle(int x1P, int y1P, int x2P, int y2P, int x3P, int y3P,int red,int green,int blue, int lineWidth)
Triangle* t01 = new Triangle(120, 450, 70, 550, 170, 550, 0, 0, 0, 0, 3);
t01->draw();
delete t01;
Triangle* t02 = new Triangle(320, 450, 270, 550, 370, 550, 0, 0, 0, 0, 3);
t02->draw();
delete t02;
Triangle* t03 = new Triangle(520, 450, 470, 550, 570, 550, 0, 0, 0, 0, 3);
t03->draw();
delete t03;
Triangle* t04 = new Triangle(720, 450, 670, 550, 770, 550, 0, 0, 0, 0, 3);
t04->draw();
delete t04;

Triangle* t11 = new Triangle(120, 450, 70, 550, 170, 550, 255, 255, 0, 3);
t11->fill();
delete t11;
Triangle* t12 = new Triangle(320, 450, 270, 550, 370, 550, 127, 255, 0, 3);
t12->fill();
delete t12;
Triangle* t13 = new Triangle(520, 450, 470, 550, 570, 550, 0, 255, 0, 3);
t13->fill();
delete t13;
Triangle* t14 = new Triangle(720, 450, 670, 550, 770, 550, 0, 255, 127, 3);
t14->fill();
delete t14;

//Text (int xP, int yP,string text, string fontType, int fontSize, int bold_1or0, int red,int green,int blue)
StringText* circle01 = new StringText(90,160, " 0,255,255 ", "Comic Sans", 13, 1,0,0,0);
StringText* circle21 = new StringText(90,180, " Türkis", "Comic Sans", 13, 1,0,0,0);
circle01->draw();
circle21->draw();
delete circle01;
delete circle21;

```

```
StringText* circle02 = new StringText(290,160, " 0,127,255 ", "Comic Sans", 13, 1,0,0,0);
StringText* circle22 = new StringText(290,180, " Hellblau", "Comic Sans", 13, 1,0,0,0);
circle02->draw();
circle22->draw();
delete circle02;
delete circle22;
StringText* circle03 = new StringText(490,160, " 0,0,255 ", "Comic Sans", 13, 1,0,0,0);
StringText* circle23 = new StringText(490,180, " Blau", "Comic Sans", 13, 1,0,0,0);
circle03->draw();
circle23->draw();
delete circle03;
delete circle23;
StringText* circle04 = new StringText(690,160, " 127,0,255 ", "Comic Sans", 13, 1,0,0,0);
StringText* circle24 = new StringText(690,180, " Violett", "Comic Sans", 13, 1,0,0,0);
circle04->draw();
circle24->draw();
delete circle04;
delete circle24;
//-----
StringText* rectagle01 = new StringText(90,360, " 255,0,255 ", "Comic Sans", 13, 1,0,0,0);
StringText* rectagle21 = new StringText(90,380, " Pink", "Comic Sans", 13, 1,0,0,0);
rectagle01->draw();
rectagle21->draw();
delete rectagle01;
delete rectagle21;

StringText* rectagle02 = new StringText(290,360, " 255,0,127 ", "Comic Sans", 13, 1,0,0,0);
StringText* rectagle22 = new StringText(290,380, " Magenta", "Comic Sans", 13, 1,0,0,0);
rectagle02->draw();
rectagle22->draw();
delete rectagle02;
delete rectagle22;

StringText* rectagle03 = new StringText(490,360, " 255,0,0 ", "Comic Sans", 13, 1,0,0,0);
StringText* rectagle23 = new StringText(490,380, " Rot", "Comic Sans", 13, 1,0,0,0);
rectagle03->draw();
rectagle23->draw();
delete rectagle03;
delete rectagle23;
//-----
StringText* rectagle04 = new StringText(690,360, " 255,127,0 ", "Comic Sans", 13, 1,0,0,0);
StringText* rectagle24 = new StringText(690,380, " Orange", "Comic Sans", 13, 1,0,0,0);
rectagle04->draw();
rectagle24->draw();
delete rectagle04;
delete rectagle24;
//-----
```

```

StringText* triangle01 = new StringText(90,560, " 255,255,0 ", "Comic Sans", 13, 1,0,0,0);
StringText* triangle21 = new StringText(90,580, " Gelb", "Comic Sans", 13, 1,0,0,0);
triangle01->draw();
triangle21->draw();
delete triangle01;
delete triangle21;

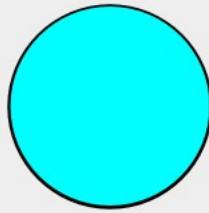
StringText* triangle02 = new StringText(290,560, " 127,255,0 ", "Comic Sans", 13, 1,0,0,0);
StringText* triangle22 = new StringText(290,580, " Grün-Gelb", "Comic Sans", 13, 1,0,0,0);
triangle02->draw();
triangle22->draw();
delete triangle02;
delete triangle22;

StringText* triangle03 = new StringText(490,560, " 0,255,0 ", "Comic Sans", 13, 1,0,0,0);
StringText* triangle23 = new StringText(490,580, " Grün", "Comic Sans", 13, 1,0,0,0);
triangle03->draw();
triangle23->draw();
delete triangle03;
delete triangle23;

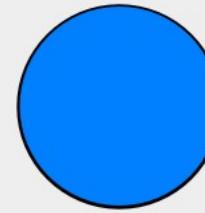
StringText* triangle04 = new StringText(690,560, " 0,255,127 ", "Comic Sans", 13, 1,0,0,0);
StringText* triangle24 = new StringText(690,580, " Mint-Grün", "Comic Sans", 13, 1,0,0,0);
triangle04->draw();
triangle24->draw();
delete triangle04;
delete triangle24;

return 0;
}

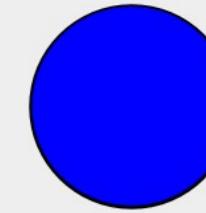
```



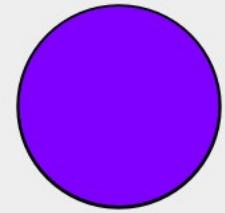
0,255,255
Türkis



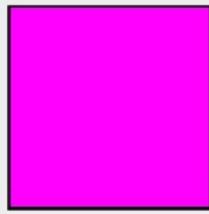
0,127,255
Hellblau



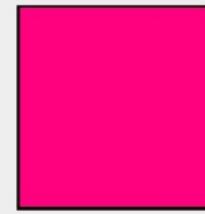
0,0,255
Blau



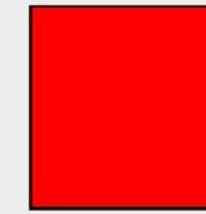
127,0,255
Violett



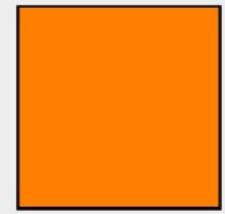
255,0,255
Pink



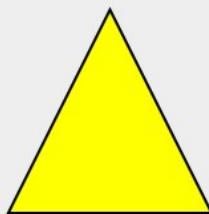
255,0,127
Magenta



255,0,0
Rot



255,127,0
Orange



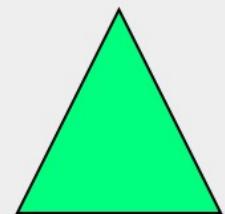
255,255,0
Gelb



127,255,0
Grün-Gelb



0,255,0
Grün



0,255,127
Mint-Grün

D.3 Anhang - Online Auftritt

Farbzauber und Funktionen

Das Kleinprojekt "Farbzauber und Funktionen", entwickelt von Mykyta Skrypnyk, Dennis Seiler und Marcus Rosengart, entstand als Teilprojekt der Projektgruppe "Programmiererlebnisse" im WS 23/24 an der Hochschule Emden / Leer.

Das Projekt wurde von Prof. Dr. Carsten Link und Frederik Gosewehr, M.Eng. betreut mit dem Ziel Programmiererlebnisse für Einsteiger zu entwickeln.

Zu diesem Zweck wurde aufbauend auf der Projektarbeit von Nurullah Damla aus dem Jahr 2022, ebenfalls betreut von Prof. Dr. Carsten Link, eine Swing-Anwendung erweitert und angepasst sowie spezifische Lehrmaterialien und Übungsaufgaben zu erstellt.

Experimentelle Einführung in die Programmierung

Die Idee für das Kleinprojekt war zunächst ähnliche Übungen einer Aufgabe aus dem Modul C/C++, ebenfalls betreut von Prof. Dr. Carsten Link, zu entwickeln, wo Schüler und Erstsemester mithilfe von Programmierstrukturen ein Bild geformt aus ASCII-Zeichen(American Standard Code for Information Interchange) auf dem Bildschirm ausgeben sollten. Zum Beispiel:

The image shows a large-scale ASCII art representation of the German flag. It is composed of a grid of characters, primarily asterisks (*), hash symbols (#), and at symbols (@). The pattern is designed to form the red, white, and black horizontal stripes of the flag. The at symbols (@) are used to represent the yellow stars on the flag's center.

Dieses Grundkonzept ist weitestgehend erhalten geblieben nur, dass die Ausgabe nicht mehr über das Terminal oder einer .txt Datei erfolgt, sondern in einer überarbeiteten Swing-Anwendung die über das Aufrufen von Funktionen in C++ oder Python das Zeichnen von geometrischen Grundformen erlaubt, freundlicherweise zur Verfügung gestellt von Prof. Dr. Carsten Link.

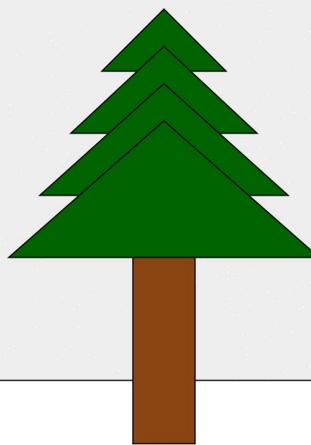
Run Stop Clear Choose

```

Mouse Position: x(797),y(535)
Fill_Rectangle: x(0),y(399), Breite(799), Hoehe(200)
Draw_Rectangle: x(0),y(399), Breite(799), Hoehe(200)
Fill_Rectangle: x(125),y(250), Breite(50), Hoehe(200)
Draw_Rectangle: x(125),y(250), Breite(50), Hoehe(200)
Fill_Triangle: x1(100),y1(150),x2(200),y2(150),x3(150),y3(100)
Draw_Triangle: x1(100),y1(150),x2(200),y2(150),x3(150),y3(100)
Fill_Triangle: x1(75),y1(200),x2(225),y2(200),x3(150),y3(130)
Draw_Triangle: x1(75),y1(200),x2(225),y2(200),x3(150),y3(130)
Fill_Triangle: x1(50),y1(250),x2(250),y2(250),x3(150),y3(160)
Draw_Triangle: x1(50),y1(250),x2(250),y2(250),x3(150),y3(160)
Fill_Triangle: x1(25),y1(300),x2(275),y2(300),x3(150),y3(190)
Draw_Triangle: x1(25),y1(300),x2(275),y2(300),x3(150),y3(190)
Text: x(250),y(0), Schriftgroesse(24)
Fill_Circle: x(550),y(375),Radius(75)
Draw_Circle: x(550),y(375),Radius(75)
Fill_Circle: x(575),y(300),Radius(50)
Draw_Circle: x(575),y(300),Radius(50)
Fill_Circle: x(600),y(250),Radius(25)
Draw_Circle: x(600),y(250),Radius(25)
Fill_Circle: x(615),y(425),Radius(15)
Draw_Circle: x(615),y(425),Radius(10)
Fill_Circle: x(615),y(375),Radius(10)
Draw_Circle: x(615),y(375),Radius(10)
Fill_Circle: x(615),y(325),Radius(10)
Draw_Circle: x(615),y(325),Radius(10)
Fill_Circle: x(610),y(265),Radius(5)
Fill_Circle: x(630),y(265),Radius(5)
Fill_Rectangle: x(575),y(250), Breite(100), Hoehe(10)
Fill_Rectangle: x(600),y(200), Breite(50), Hoehe(50)
Fill_Triangle: x1(025),y1(280),x2(675),y2(290),x3(600),y3(285)
Line: x1(575),y1(340),x2(530),y2(300)
Line: x1(670),y1(340),x2(720),y2(300)

```

Frohe Weihnachten



D.4 Anhang - Lizenz

Farbzauber und Funktionen is a combination of a Java:tm: application, editable C++ Code and exercises designed to facilitate educational exploration and inspire students to begin their coding journey. Copyright (C) 2024 Mykyta Skrypnyk, Dennis Seiler, Marcus Rosengart

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see <http://www.gnu.org/licenses/>.

E Anhang - Programmieren lernen mit dem MIT App Inventor

E.1 Schüler PDF

E.1.1 Einleitung

Willkommen Schüler und Schülerinnen! Heutzutage besitzen die meisten jungen Leute ein Smartphone auf denen ihr viele verschiedene Apps installieren könnt. Heute wollen wir euch zeigen wie ihr mit dem MIT App Inventor Programm eure eigene To Do ListApp entwickelt und wie ihr diese App sogar auf eurem eigenen Handy testen könnt.

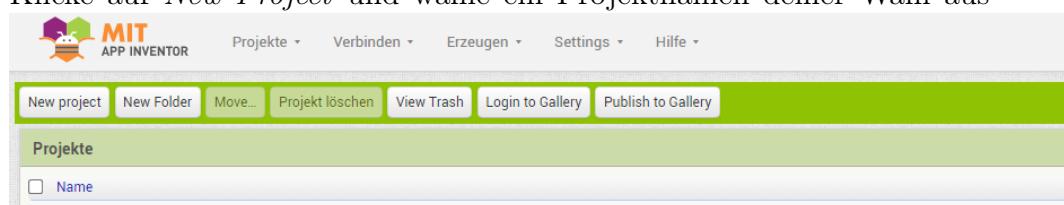
E.1.2 Lernziele:

- Verstehen, wie man einfache Apps mit dem MIT App Inventor erstellt
 - Oberfläche (Designer)
 - Programmierung (Blöcke)
- Programmierkonzepte wie Schleifen, Listen und Indizes kennenzulernen
- Abschluss: Versuchen eine eigene *To-Do List* App zu erstellen

E.1.3 Vorbereitung

E.1.3.1 Neues Projekt erstellen

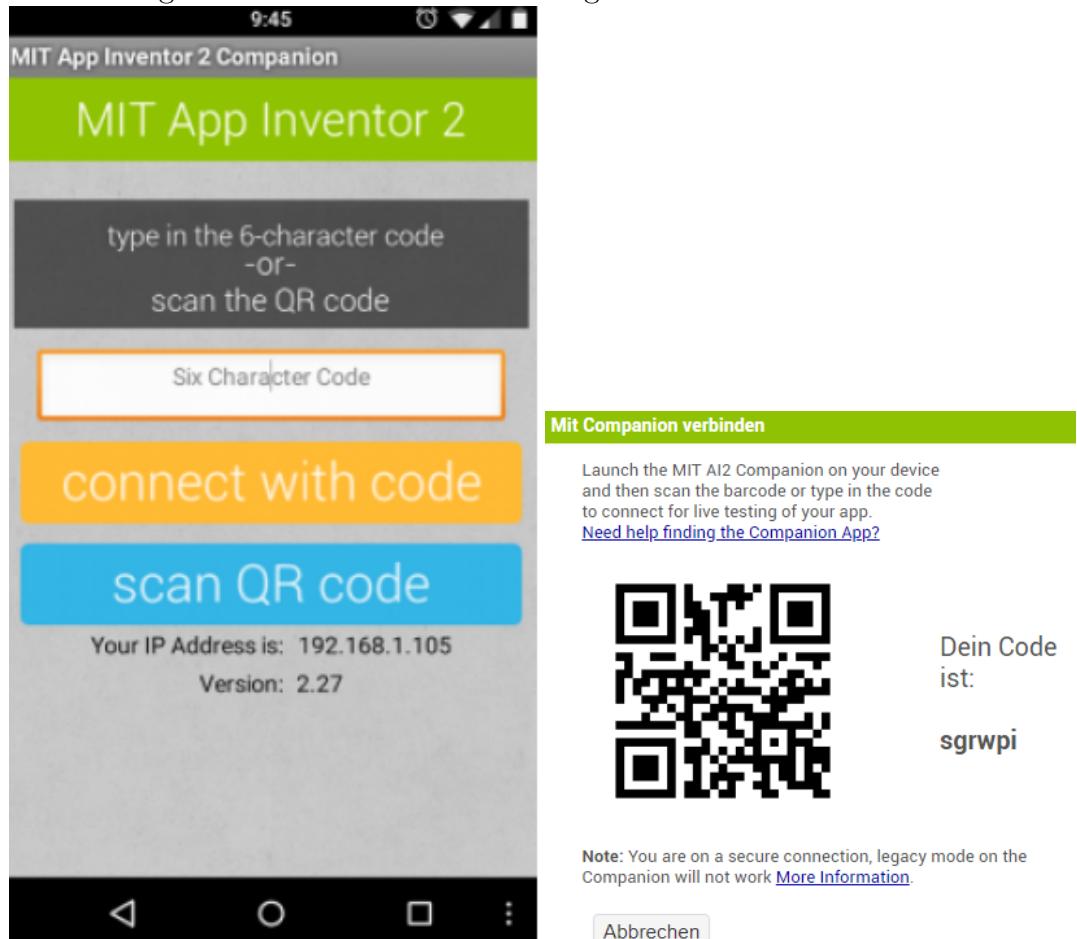
- Klicke auf *New Project* und wähle ein Projektnamen deiner Wahl aus



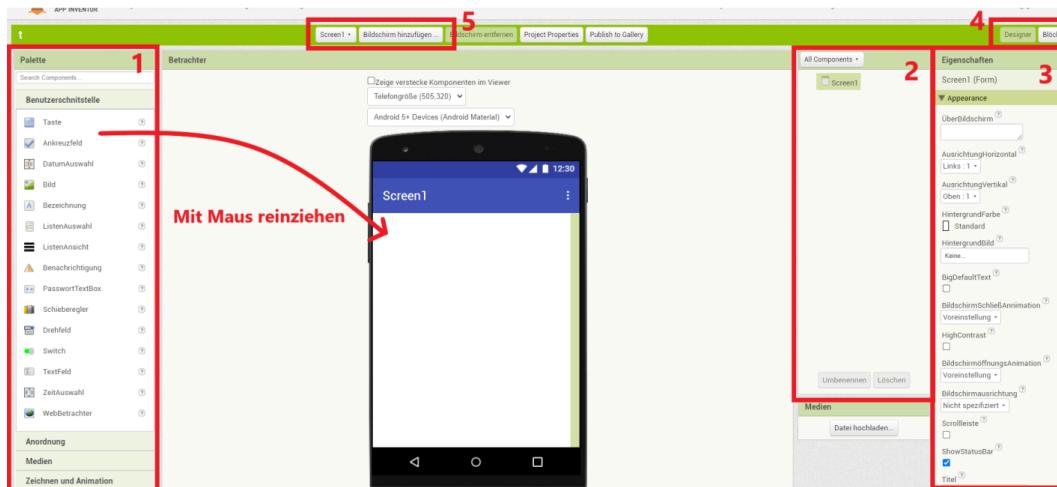
E.1.3.2 Verbinde dich mit deinem Smartphone

Du kannst mit deinem Smartphone (oder Emulator) live dabei zusehen wie deine App aussieht und diese direkt testen

- Downloade und installiere die *MIT AI2 Companion App* auf deinem Smartphone
- Auf dem PC wähle Verbinden -> AI Companion
- Du kannst mit deinem Smartphone entweder den QR code scannen oder den 6-stelligen Code in die Textbox eintragen



E.1.4 Aufgabe 0 - Benutzeroberfläche kennenlernen (ca. 5 Minuten)



1. **Palette:** Komponenten wie Textfelder, Tasten, etc. die du zu deiner App hinzufügen kannst
2. **Komponenten:** Alle derzeit genutzten Komponenten in deinem Screen werden hier aufgelistet
3. **Eigenschaften:** Klicke eine Komponente in (2) an und ändere die Eigenschaften wie z.B. Ausrichtung, Text, und mehr
4. **Designer/Blöcke:** Wechsel zwischen der Designer (Oberfläche) und Blöcke (Programmieren) Ansicht
5. **Screens/Bildschirme:** Hinzufügen und wechsel zwischen mehreren Bildschirmen (wichtig für die Aufgaben)

E.1.5 Aufgabe 1 - Bedingungen (ca. 10 Minuten)



Mit Bedingungen kannst du der App folgendes sagen:

Wenn etwas passiert, **dann** mache etwas Bestimmtes; **sonst** mache etwas anderes.

Bedingungsblöcke sind ein essentieller Teil der Programmierung und wird auch für unsere To Do List-App wichtig sein.

In der ersten Aufgabe werden wir mithilfe des *wenn-dann* Blocks

- die Hintergrundfarbe des Bildschirms abhängig von der gedrückten Taste ändern.

1.1 Oberfläche(Designer)

- Versuche diese Oberfläche im Designer nachzubauen



- Hier sind die Komponenten die zu Screen1 hinzugefügt werden



- 3 Tasten
 - * Taste_rot
 - * Taste_blau
 - * Taste_grün
 - 1 Horizontale Ausrichtung
- Benenne die Komponenten so wie oben im Bild um. Aussagekräftige Namen werden dir beim Programmieren sehr helfen!

- Hinweis: Du kannst mit *AusrichtungHorizontal* und *AusrichtungVertikal* die Felder in der Mitte platzieren

1.2 Programmierung(Blöcke) Wir haben nun zwar die 3 Tasten erstellt. Wenn man jedoch auf die Tasten klickt, dann ändert sich die Hintergrundfarbe aber nicht? Das liegt daran, dass du noch nicht festgelegt hast was passieren soll sobald man auf einer der Tasten wie z.B. *grün* klickt. Dazu wechseln wir von der Designer-Ansicht zur Blöcke-Ansicht.

- Deine Aufgabe
 - Wenn *rot*-Taste geklickt wird, dann setze die Hintergrundfarbe zu *rot*
 - Wenn *grün*-Taste geklickt wird, dann setze die Hintergrundfarbe zu *grün*
 - Wenn *blau*-Taste geklickt wird, dann setze die Hintergrundfarbe zu *blau*

- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



- Probiere deine App auf deinem Smartphone/Emulator aus

E.1.6 Aufgabe 2 - Listen erstellen und anzeigen (ca. 10 Minuten)

Was ist das wichtigste für eine To Do Liste? Richtig, wir brauchen unbedingt Listen in der wir uns unsere Aufgaben notieren können. In der Informatik musst man sich Listen so vorstellen, als wäre sie eine Tabelle mit zwei Spalten.

Listen-Index	Eintrag
1	
2	
3	
4	

- Wenn du etwas neues zur Liste hinzufügen willst wird der Eintrag in das nächste freie Feld eingetragen. Außerdem fängt man in der Informatik in der Regel an mit 0 anstatt 1 an zu zählen.

Listen-Index	Eintrag
1	Aufwachen
2	Frühstücke
3	Zur Schule gehen
4	Mittagessen

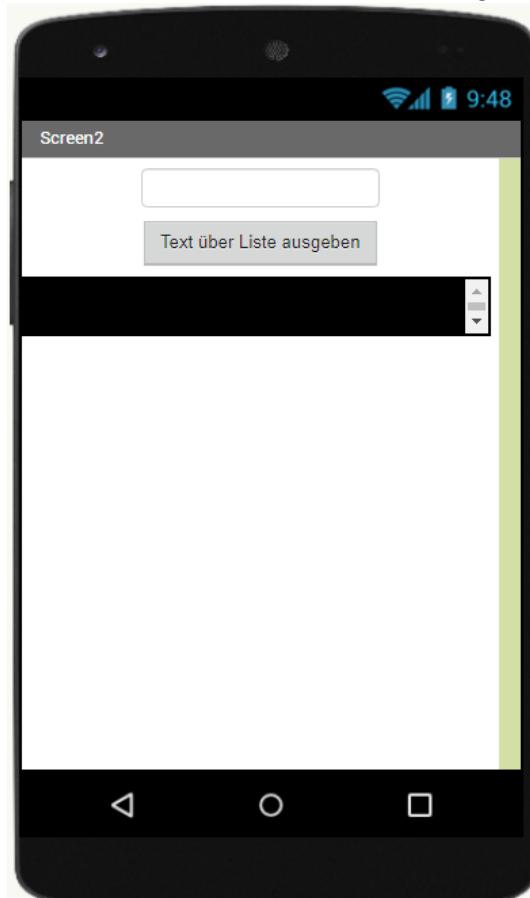
Im Folgenden lernst du wie du

- eine neue Liste erstellst
- und diese Liste anzeigen kannst

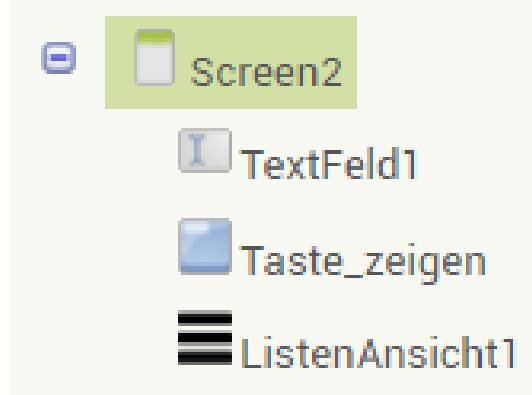
2.1 Oberfläche(Designer)

Wichtig: Füge ein neuen Bildschirm *Screen2* hinzu und wechsle darauf

- Versuche diese Oberfläche im Designer nachzubauen



- Hier sind die Komponenten die zu Screen2 hinzugefügt werden



- 1 Textfeld
- 1 Taste
- 1 Listen Ansicht

2.2 Programmierung(Blöcke)

- Deine Aufgabe
 - initialisiere/erstelle eine leere Liste
 - sobald auf die Taste geklickt wird, soll der Inhalt vom Textfeld zur Listenansicht hinzugefügt und angezeigt werden
- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



- Probiere deine App auf deinem Smartphone/Emulator aus

E.1.7 Aufgabe 3 - Einträge aus Liste auswählen/löschen (ca. 10 Minuten)

Erinnerst du dich daran wie eine einfache Liste in der Informatik aussieht?

Listen-Index	Eintrag
1	Aufwachen

Listen-Index	Eintrag
2	Frühstücken
3	Zur Schule gehen
4	Frühstücken

In dieser Liste gibt es einen doppelten Eintrag *Frühstücken*. Wie können wir sicher sein, dass wir den richtigen *Frühstücken* Eintrag ausgewählt haben? Da die Einträge durchnummieriert sind verwendet man häufig die Stelle bzw. den Index um Einträge innerhalb einer Liste auswählen. Anstatt also zu versuchen den Eintrag *Frühstücken* zu finden würden wir dem Computer sagen, dass er nach dem Eintrag an der Stelle (2) in der Liste suchen soll.

In der folgenden Aufgabe wirst du lernen

- wie man einen Listenelement anhand des Indizes löscht
- die Listenansicht aktualisiert

3.1 Oberfläche(Designer)

Wichtig: Füge ein neuen Bildschirm *Screen3* hinzu und wechsle darauf

- Versuche diese Oberfläche im Designer nachzubauen



- Hier sind die Komponenten die zu Screen3 hinzugefügt werden

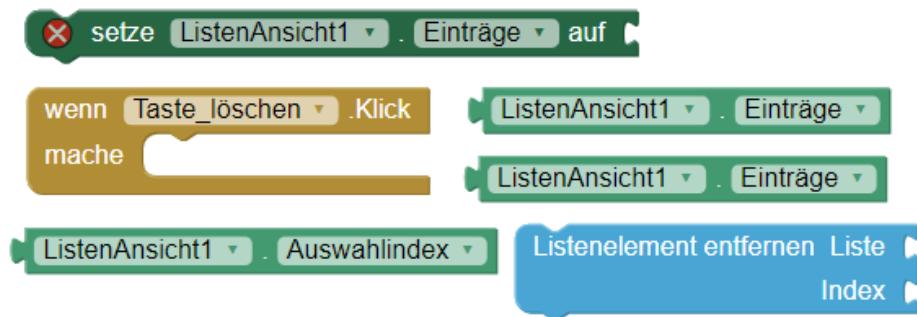


- 1 Taste
- 1 Listen Ansicht
 - Du kannst auf der rechten Seite unter *Behaviour* die Einträge hinzufügen

3.2 Programmierung (Blöcke)

- Deine Aufgabe
 - Entferne das Listenelement aus den Einträgen der Listen-Ansicht
 - Aktualisiere die Listenansicht mit den neuen Einträgen

- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



- Probiere deine App auf deinem Smartphone/Emulator aus

E.1.8 Aufgabe 4 - Listen permanent speichern (TinyDatenBank) (ca. 10 Minuten)



Ihr habt nun gelernt wie man Einträge zu einer Liste hinzufügt und entfernt. Das Problem jedoch besteht zurzeit, dass sobald ihr eure App schließt eure To Do Liste ihr gesamten Einträge vergisst und somit beim nächsten Start wieder leer ist. Solltet ihr also eure Liste stattdessen in einem Speichermedium wie z.B. eine TinyDatenbank speichern, dann bleibt eure To Do Liste auch nach einem App Neustart bestehen. Der *Tag* bzw. Kennzeichen ist lediglich die Bezeichnung unter welchen Namen der Wert gespeichert werden soll.

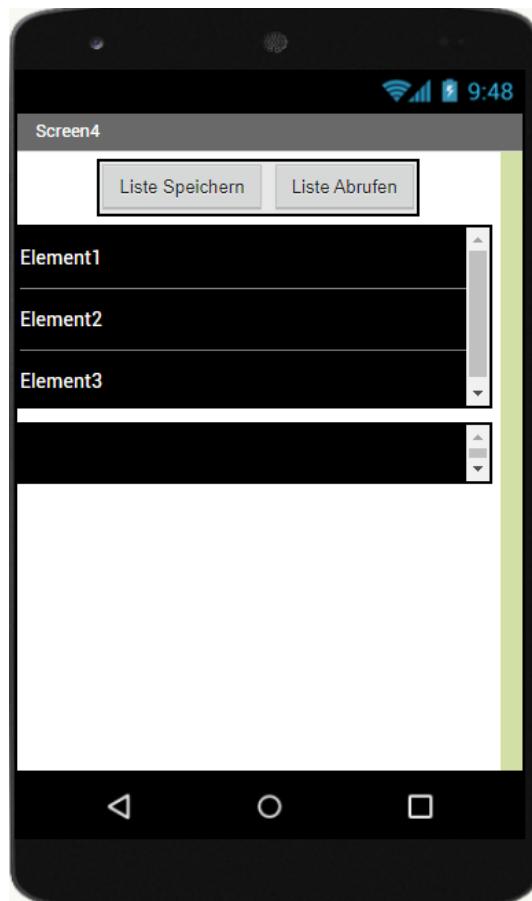
In der folgenden Aufgabe wirst du lernen

- wie man die ListenAnsicht in der Tinydatenbank speichert und abruft

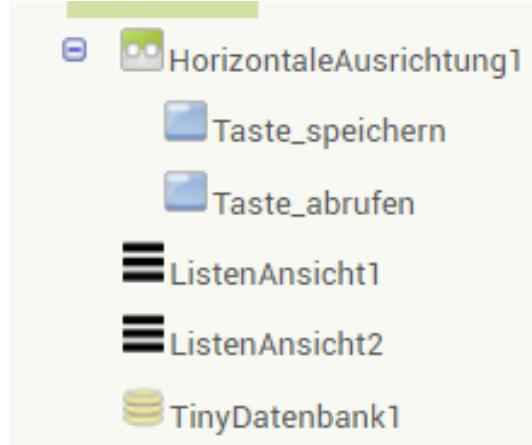
4.1 Oberfläche (Designer)

Wichtig: Füge ein neuen Bildschirm *Screen4* hinzu und wechsle darauf

- Versuche diese Oberfläche im Designer nachzubauen



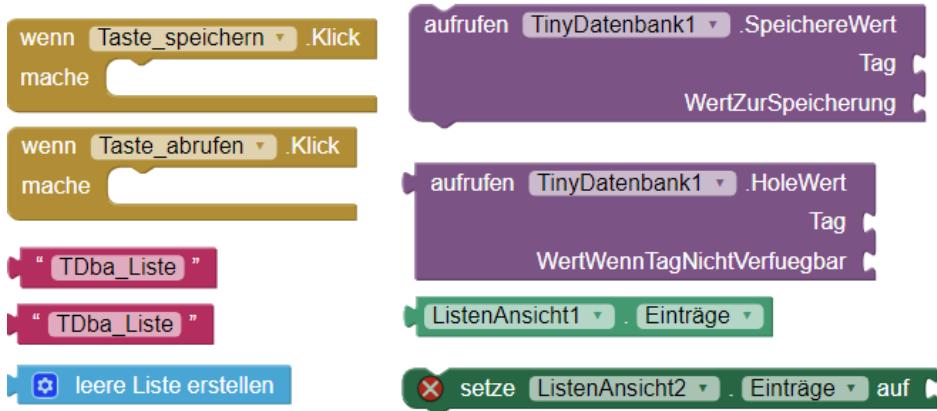
- Hier sind die Komponenten die zu Screen4 hinzugefügt werden



- 1 Horizontale Ausrichtung
- 2 Tasten
- 2 Listen Ansichten
 - ListenAnsicht1 hat vorhandene Elemente
 - ListenAnsicht2 ist leer
- 1 TinyDatenbank (unter Speicher)

4.2 Programmierung(Blöcke)

- Deine Aufgabe
 - Wenn auf *Liste Speichern* geklickt wird, soll die Listenansicht1 in der TinyDatenbank gespeichert werden
 - Wenn auf *Liste Abrufen* geklickt wird, soll der Wert aus der Tiny-Datenbank geholt werden und in Listenansicht 2 angezeigt werden
- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



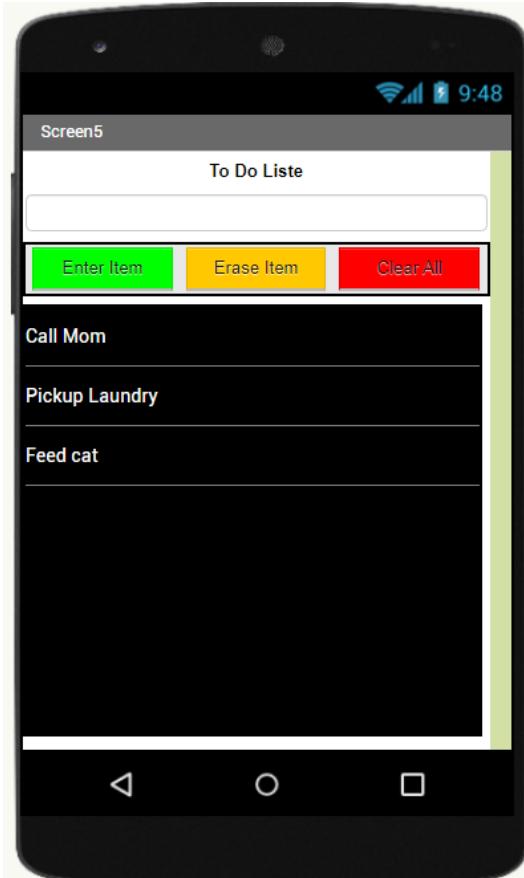
- Probiere deine App auf deinem Smartphone/Emulator aus

E.1.9 Abschlussaufgabe - To Do List App (ca. 30 Minuten)

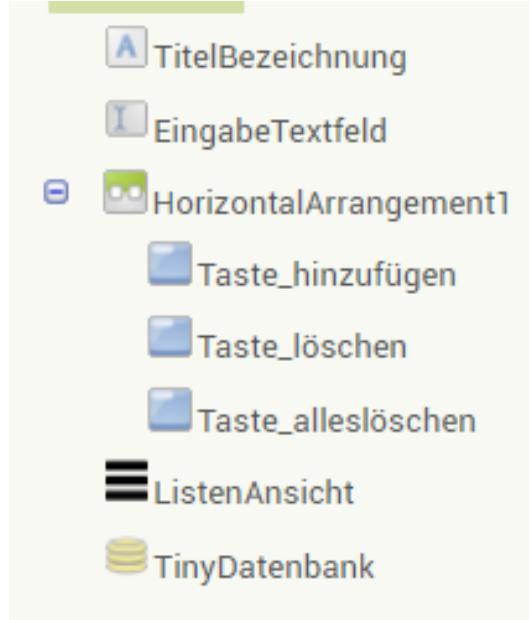
Ihr habt nun grundlegende Sachen über Bedingungen, Listen und Speicher kennengelernt. Nun gilt es die Sachen aus Aufgabe 1-4 anzuwenden und eure eigene To Do List App zu erstellen. Hier ist ein Beispiel wie eine einfache To Do List App aussehen könnte, aber ihr könnt natürlich eure Kreativität freien lauf lassen!

Wichtig: Füge ein neuen Bildschirm *Screen5* hinzu und wechsle darauf

5.1 Oberfläche (Designer)



- Hier sind die Komponenten die zu Screen4 hinzugefügt werden



4.2 Programmierung (Blöcke)

Deine Aufgaben bestehen aus mehreren Teilaufgaben

- Wenn der Bildschirm/Screen aufgerufen wird, dann

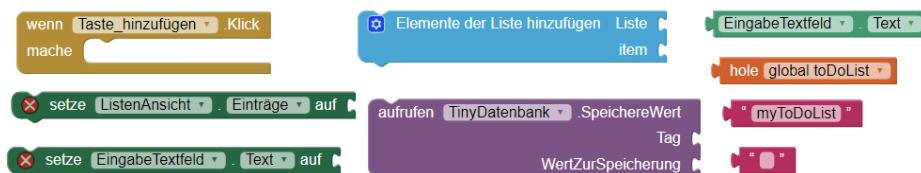
global initialisieren [ToDoList] auf leere Liste erstellen

- Hole das *myToDoList* Kennzeichen von der TinyDatenbank und speichere den Wert in die *ToDoList*
- Aktualisiere die Listenansicht mit dem Inhalt aus *ToDoList*
- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



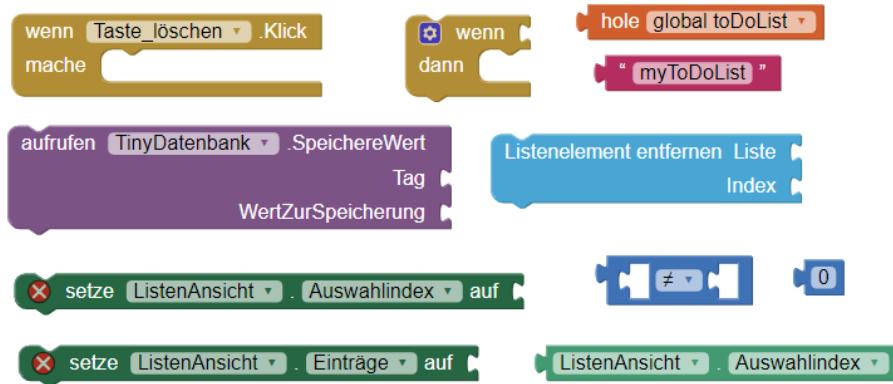
- Wenn auf *Enter Item* geklickt wird,

- füge den Eintrag aus dem Textfeld der Liste *ToDoList* hinzu
- aktualisiere die TinyDatenbank, damit der Speicher den neuesten Stand besitzt
- aktualisiere die Listenansicht
- leere das Textfeld
- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



- Wenn auf *Erase Item* geklickt wird,

- überprüfe zunächst die Bedingung ob der Benutzer in der Listenansicht tatsächlich ein Eintrag ausgewählt hat (Index ungleich 0)
- Falls ja, entferne den Eintrag aus der *ToDoList*
- aktualisiere die ListenAnsicht
- aktualisiere den Speicher aus der TinyDatenbank unter der Bezeichnung/Tag *myToDoList*
- setz den Index von der Listenansicht zurück (Index gleich 0)
- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



- Wenn auf *Clear All* geklickt wird,

- setze die *toList* zu einer leeren Liste
- setze die Listenansicht zu einer leeren Liste
- Lösche alle Speichereinträge unter dem Kennzeichen/Tag *myToDoList*
- Hier sind die Blöcke die du brauchst um diese Aufgabe zu lösen



E.2 Lehrer PDF

E.2.1 Installationsanleitung

Man kann entweder die Entwicklungsumgebung über die offizielle Webseite (<https://appinventor.mit.edu/>) starten oder wie im Folgenden die Entwicklungsumgebung lokal auf dem eigenen Rechner installieren.

Sollte die Entwicklungsumgebung über die VM (Vagrant) aus irgendeinem Grund nicht für anderen Rechner im Labor erreichbar sein, wird empfohlen, die Installation mithilfe von Vagrant auf jedem einzelnen Rechner zu tätigen und ggf. mit einem Shell-Skript zu automatisieren.

E.2.1.1 Manuelle Installation

Die manuelle Installationsanleitung für die MIT App Inventor Umgebung und die Anforderungen für die verschiedenen Betriebssysteme kann man aus der [offiziellen Dokumentation](#) entnehmen, empfohlen wird jedoch die Installation mithilfe von Vagrant.

E.2.1.2 Installation mit Vagrant (Virtualbox)

Eine allgemeine Anleitung findet man auf der GitHub-Repository ([hier](#)). Benötigt werden

- Java Development Kit (JDK) 1.8
- Apache Ant 1.10+

MacOS

1. Virtualbox Version 7.0.14 und Vagrant installieren
2. Security Restrictions in MacOS für Virtualbox anpassen (siehe [Link](#))
3. Klone die Repository mit
`git clone https://github.com/haifrosch/appinventor-sources-master.git`
4. Dann folgende Befehle im `|appinventor-sources-master` Verzeichnis ausführen
`vagrant plugin install vagrant-vbguest # only once`
`vagrant up # initializes the VM`
Beim ersten Mal wird es einige Minuten dauern, um die Abhängigkeiten für die VM einzurichten.
5. Um sich mit der VM zu verbinden `vagrant ssh`
6. Dependencies updaten mit `git submodule update --init`

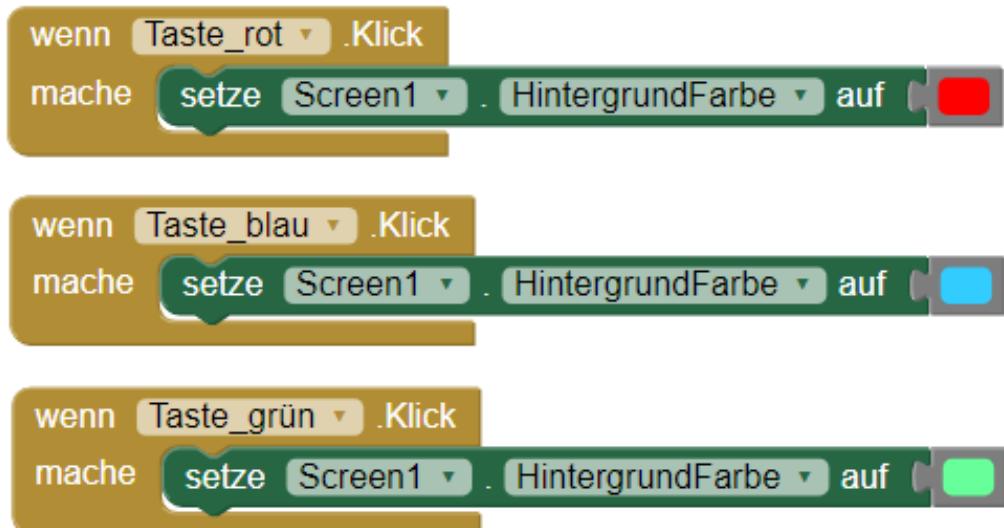
7. Dann bauen mit *ant*
8. Und mit dem folgenden Befehl den Server starten *start_appinventor*
9. Die Anwendung ist nun über *localhost:8888* erreichbar
10. Einloggen kann man sich mit einem beliebigen Benutzernamen über den Google Sign-In

Mit *vagrant halt* und *vagrant destroy* lässt sich die VM anhalten/löschen.

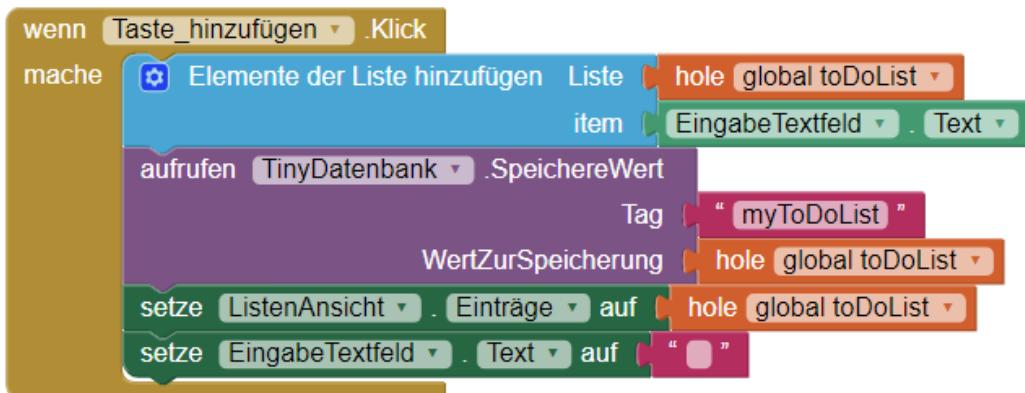
Windows/Linux Die Installation ist bis auf Schritt (2) identisch mit der Anleitung für MacOS.

E.2.2 Musterlösungen

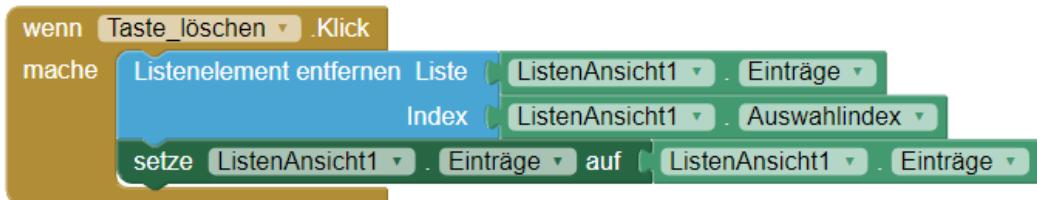
E.2.2.1 Musterlösung zu 1.2



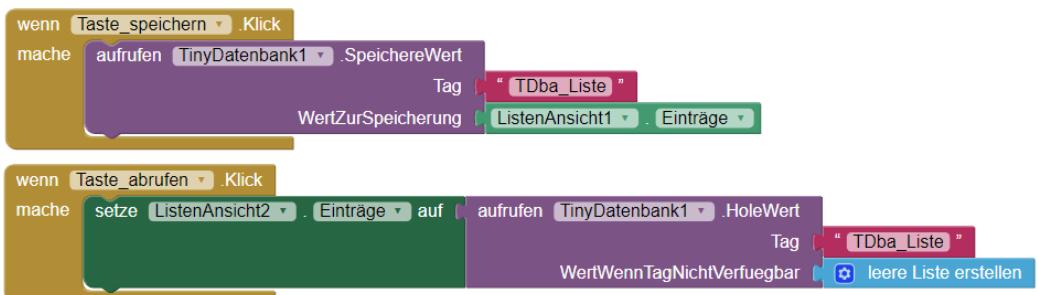
E.2.2.2 Musterlösung zu 2.2



E.2.2.3 Musterlösung zu 3.2



E.2.2.4 Musterlösung zu 4.2



E.2.2.5 Musterlösung zu 5.2

