

# Inhaltsverzeichnis

<b>1 Zielbestimmung</b>	<b>7</b>
1.1 Musskriterien . . . . .	7
1.2 Sollkriterien . . . . .	8
1.3 Kannkriterien . . . . .	9
1.4 Abgrenzungskriterien . . . . .	11
<b>2 Produktübersicht</b>	<b>12</b>
2.1 Betriebsbedingungen . . . . .	12
2.2 Use-Cases . . . . .	13
2.3 Aktivitäten . . . . .	18
<b>3 Produktfunktionen</b>	<b>22</b>
3.1 Welt-Funktionen . . . . .	22
3.2 Block-Funktionen . . . . .	24
3.3 Item-Funktionen . . . . .	28
<b>4 Nichtfunktionale Anforderungen</b>	<b>31</b>
4.1 Funktionalität . . . . .	31
4.2 Sicherheit . . . . .	31
4.3 Benutzbarkeit . . . . .	32
4.4 Änderbarkeit . . . . .	32
4.5 Nichtfunktionale Produktanforderungen . . . . .	33
<b>5 Benutzeroberfläche</b>	<b>34</b>
5.1 Lerncomputer in der Minecraft-Welt . . . . .	34
5.2 Brillen-Item . . . . .	35
5.3 Register-Block . . . . .	35
5.4 Speicher-Block . . . . .	36
5.5 Programmier-Block . . . . .	37
5.6 Terminal-Block . . . . .	38
5.7 Konfigurierbaren Befehlssatz-Item . . . . .	38
5.8 Steuerwerk-Block . . . . .	39
5.9 Taktgeber-Block . . . . .	39
5.10 Handbuch . . . . .	40
<b>6 Technische Produktumgebung</b>	<b>41</b>
6.1 Software . . . . .	41

6.2 Hardware . . . . .	41
<b>7 Glossar</b>	<b>42</b>

## Abbildungsverzeichnis

2.1	Use-Case-Diagramm: Minecraft-Welt . . . . .	13
2.2	Use-Case-Diagramm: Programmier-Block . . . . .	14
2.3	Use-Case-Diagramm: Steuerwerk-Block . . . . .	15
2.4	Use-Case-Diagramm: Taktgeber-Block . . . . .	15
2.5	Use-Case-Diagramm: Speicher-Block . . . . .	16
2.6	Use-Case-Diagramm: Standard-Minecraft-Block . . . . .	17
2.7	Aktivitätsdiagramm Gesamt . . . . .	18
2.8	Aktivitätsdiagramm Steuerwerk . . . . .	19
2.9	Aktivitätsdiagramm Programmier-Block . . . . .	20
2.10	Aktivitätsdiagramm Speicher-Block . . . . .	21
5.1	Ansicht des MIMA-Computeraufbaus in Minecraft . . . . .	34
5.2	GUI des Brillenitems . . . . .	35
5.3	GUI des Register-Blocks . . . . .	35
5.4	GUI des Speicherblocks . . . . .	36
5.5	GUI des Programmier-Blocks . . . . .	37
5.6	GUI des Programmier-Blocks mit Hilfefenster . . . . .	37
5.7	GUI des Terminal-Blocks . . . . .	38
5.8	GUI des konfigurierbaren Befehlssatz-Items . . . . .	38
5.9	GUI des Steuerwerk-Blocks . . . . .	39
5.10	GUI des Taktgeber-Blocks . . . . .	39
5.11	GUI des Handbuches . . . . .	40

# 1 Zielbestimmung

Das Erlernen der Grundlagen von Rechnerarchitekturen ist ein elementarer Bestandteil des Studiums der Informatik und Informationstechnik sowie oftmals in den Grundzügen bereits in der schulischen Oberstufe. Dieses Wissen ist in Anbetracht der Häufigkeit, mit der im Alltag Begegnungen mit Computern auftreten, zeitgemäß und wichtig. Nichtsdestotrotz ist die Lehre in diesem Themenbereich oft höchst theoretisch und trocken, da auf diesem sehr Hardware-nahen Level kaum Möglichkeiten bestehen, unter Nutzung von visuellem Feedback auszuprobieren und zu testen. Die Minecraft-Modifikation, die im Rahmen dieses Projektes entwickelt wird, wird an genau dieser Stelle Abhilfe schaffen. Das Produkt bietet die Möglichkeit, in einer weithin bekannten Spielumgebung spielerisch den Aufbau und die Programmierung der mikroprogrammierten Minimalmaschine sowie eines Prozessors mit RISC-V-Architektur zu üben. So kann ohne Gefahr, Elektronik zu beschädigen, erlernt werden, welcher Befehl wie funktioniert, welche Funktion welches Bauteil hat und wie man die entsprechenden Assembler-Befehlssätze zum Schreiben vollständiger Programme nutzt. Besonders hervorzuheben ist dabei der Produktfokus auf der Visualisierung der Vorgänge im durch den Nutzer entworfenen Lerncomputer, der es erlaubt, alle Vorgänge im Rechner einzusehen.

## 1.1 Musskriterien

⟨RM1⟩ Die Mod muss im Kreativmodus von Minecraft verwendet werden können.

⟨RM2⟩ Die Mod muss den Bau eines Lerncomputers ermöglichen, anhand dessen der Nutzer Rechnerarchitektur lernen kann.

⟨RM3⟩ Die Mod muss eine MIMA-Architektur nach Tamim Asfour simulieren können.

⟨RM4⟩ Die Mod muss vom Nutzer eingegebenen MIMA-Assembler-Code ausführen können.

⟨RM5⟩ Es muss ein Block-Typ für Busse, die ALU, den Taktgeber, den Hauptspeicher, das Steuerwerk, die Register und den Programmeditor existieren. Die Blöcke müssen die Standard-Minecraft-Blockfunktionen wie Platzieren, Zerstören, Explodieren und das Hinterlassen von Drops unterstützen.

*(RM6)* Der Speicher-Block muss ein Programm-Item aufnehmen können und so Programme in den Speicher des Lerncomputers laden.

*(RM7)* Der Steuerwerk-Block muss ein Befehlssatz-Item aufnehmen können und so den aktiven Befehlssatz festlegen.

*(RM8)* Der Programmier-Block muss ein Programm auf ein Programm-Item schreiben können. Dazu müssen das Programm-Item und ein Befehlssatz-Item im Programmier-Block liegen.

*(RM9)* Der Programmier-Block muss über eine GUI mit Text-Editor für die Programmeingabe durch den Nutzer verfügen.

*(RM10)* Es muss ein Item für die Speicherung und den Transport von Programmcode, für jeden angebotenen Befehlssatz und für ein ins Spiel integriertes Handbuch existieren.

*(RM11)* Der Taktgeber-Block muss einen schrittweisen Modus anbietet, bei dem durch Interaktion der jeweils nächste Simulationstakt ausgelöst werden kann. Eine solche Interaktion löst auch den Programmstart aus.

## 1.2 Sollkriterien

*(RS1)* Der Taktgeber-Block soll die Festlegung verschiedener Taktfrequenzen für den Lerncomputer ermöglichen.

*(RS2)* Der Register-Block soll eine GUI zur Festlegung seiner Funktion in der Architektur haben.

*(RS3)* Der Taktgeber soll den Takt visualisieren.

*(RS4)* Der Speicher-Block soll eine GUI zur Inspizierung des aktuellen Speicherinhaltes bieten.

*(RS5)* Es soll ein Brillen-Item existieren, dessen Tragen es erlaubt, bei Betrachtung eines Lerncomputer-Blocks Statusanzeigen mit technischen Informationen zu erhalten.

⟨RS6⟩ Der Steuerwerk-Block soll eine GUI zur Erklärung der angebotenen Befehlssätze und dafür benötigten Bauteile, sowie ob diese bereits angebunden sind, anbieten.

⟨RS7⟩ Die Bus-Blöcke sollen von außen sichtbar visualisieren, ob und welche Art von Daten auf dem Bus präsent sind.

⟨RS8⟩ Die Mod soll eine RISC-V-Architektur simulieren.

⟨RS9⟩ Die Mod soll vom Nutzer eingegebenen RISC-V Assembler-Code ausführen können.

⟨RS10⟩ Es sollen alle relevanten Bauteile des RISC-V-Prozessors durch Minecraft-Blöcke dargestellt werden.

⟨RS11⟩ Der ALU-Block soll die Rechenaktivität äußerlich visualisieren.

⟨RS12⟩ Der Speicher-Block und die Register sollen Zugriffe äußerlich visualisieren.

### 1.3 Kannkriterien

⟨RC1⟩ Das Programmier-Block-GUI kann ein optionales Fenster mit Tipps haben. Um diese Funktion bereitzustellen muss ein Befehlssatz-Item im Programmier-Block liegen.

⟨RC2⟩ Die Bus-Blöcke können ihr Aussehen anpassen, genauer die Lage ihrer Anschlüsse. Diese können sie an umliegenden Blöcken anpassen, um die Verbindung zu visualisieren.

⟨RC3⟩ Die Items und Blöcke können Rezepte haben, sodass man sie auch im Überlebensmodus erhalten kann.

⟨RC4⟩ Es kann einen PAUSE-Befehl geben, den der Nutzer im Programmcode einbauen kann und welcher die Simulation in den schrittweisen Modus übergehen lässt.

⟨RC5⟩ Es kann ein Redstone-Interface-Block existieren, der als zusätzliches Register mit Ein- und Ausgabemöglichkeit im Minecraft-Redstone-System agiert.

⟨RC6⟩ Der Programmier-Block kann Beispielprogramme enthalten.

⟨RC7⟩ Es kann ein Terminal-Block existieren, der mithilfe eines ansprechbaren Registers Text anzeigen kann.

⟨RC8⟩ Das Steuerwerk kann über eine äußere Visualisierung seines Zustandes verfügen.

⟨RC9⟩ Der Taktgeber kann einen Echtzeit-Modus haben, der Taktfrequenzen oberhalb der Standard-Minecraft-Ticks erlaubt.

⟨RC10⟩ Es kann ein Wireless-Register-Block existieren, der seinen Registerwert mit Wireless-Register-Blöcken anderer Computer in der Minecraft-Welt synchronisieren kann.

⟨RC11⟩ Der Terminal-Block kann ein auslesbares Register haben, in das der Benutzer Text eingeben kann.

⟨RC12⟩ Es kann ein konfigurierbares Befehlssatz-Item existieren, mit dem neue MIMA-Befehle mit Mikrocode definiert werden können. Mit diesem kann unter anderem der Aufbau der MIMA modifiziert werden.

⟨RC13⟩ Das Programmier-Block-GUI kann eine einfache MIMA-/RISC-V-Assembler-Syntax-Unterstützung anbieten. Um diese Funktion bereitzustellen muss ein Befehlssatz-Item im Programmier-Block liegen.

⟨RC14⟩ Es kann eine GUI für das konfigurierbare Befehlssatz-Item existieren, die eine einfache und fehlerfreie Bearbeitung erleichtert.

⟨RC15⟩ Es kann ein konfigurierbares Befehlssatz-Item existieren, das Assembler-Code, Anzahl der Register, Busbreite und die Größe des Speichers spezifiziert.

⟨RC16⟩ Der Programmier-Block kann den Dateiimport aus Textdateien unterstützen.

## 1.4 Abgrenzungskriterien

⟨RW1⟩ Die Mod bietet eine performance-orientierte Simulation.

⟨RW2⟩ Der Festplattenspeicher des Computers, auf dem Minecraft mit der Mod läuft, wird als limitiert betrachtet.

⟨RW3⟩ Die Mod bietet alle GUIs und Datenanzeigen in Sprachen, außer Deutsch und Englisch, an.

⟨RW4⟩ Die Mod bietet eine Minecraft Bedrock Version Unterstützung.

⟨RW5⟩ Der in Minecraft simulierte Lerncomputer benötigt eine Minecraft-Stromversorgung.

## 2 Produktübersicht

### 2.1 Betriebsbedingungen

- privater oder im Lehrbetrieb genutzter PC oder Laptop
- keine Betriebszeit abseits der aktiven Nutzungszeit

## 2.2 Use-Cases

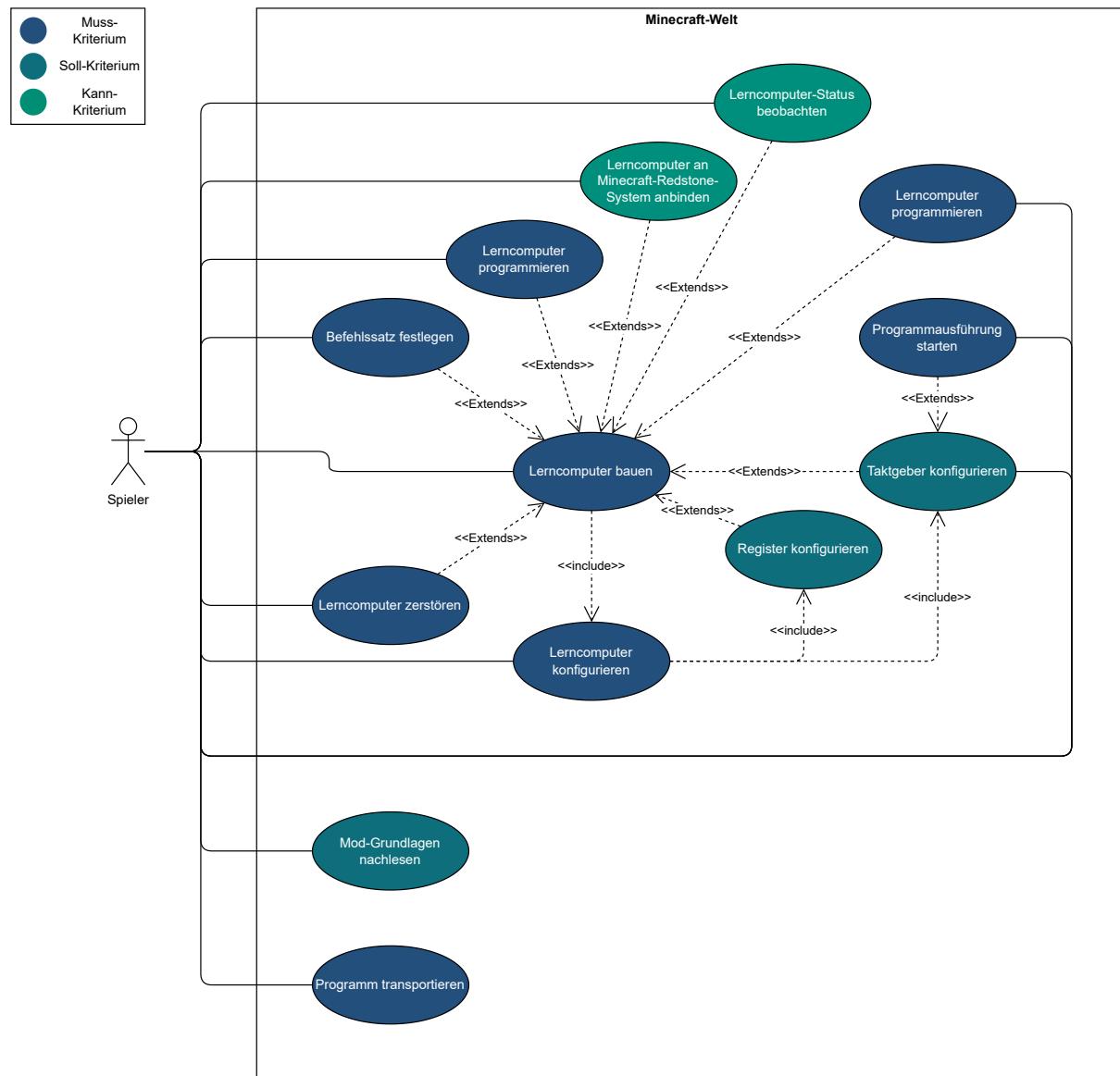


Abbildung 2.1: Use-Case-Diagramm: Minecraft-Welt

Abbildung 2.1 zeigt einen Überblick über die Anwendungsmöglichkeiten der Mod. Der Akteur ist hier ein Spieler in der Minecraft-Welt. Die Abläufe der Use-Cases sind detailliert im Aktivitätsdiagramm in Abbildung 2.7 dargestellt.

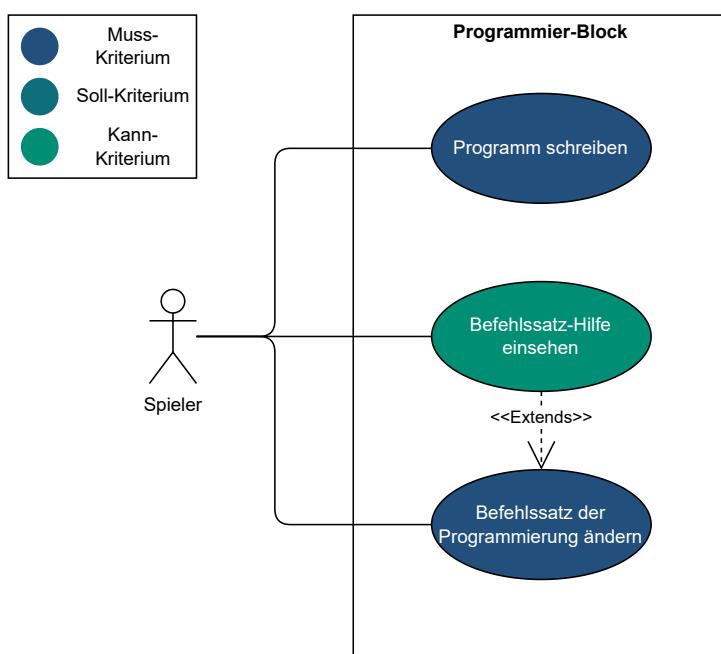


Abbildung 2.2: Use-Case-Diagramm: Programmier-Block

Abbildung 2.2 zeigt ein Use-Case-Diagramm für den Programmier-Block. Der Akteur ist hier ein Spieler, welcher ein Programm bearbeiten möchte. Der Ablauf von Interaktionen mit dem Programmier-Block wird im Aktivitätsdiagramm in Abbildung 2.9 dargestellt.

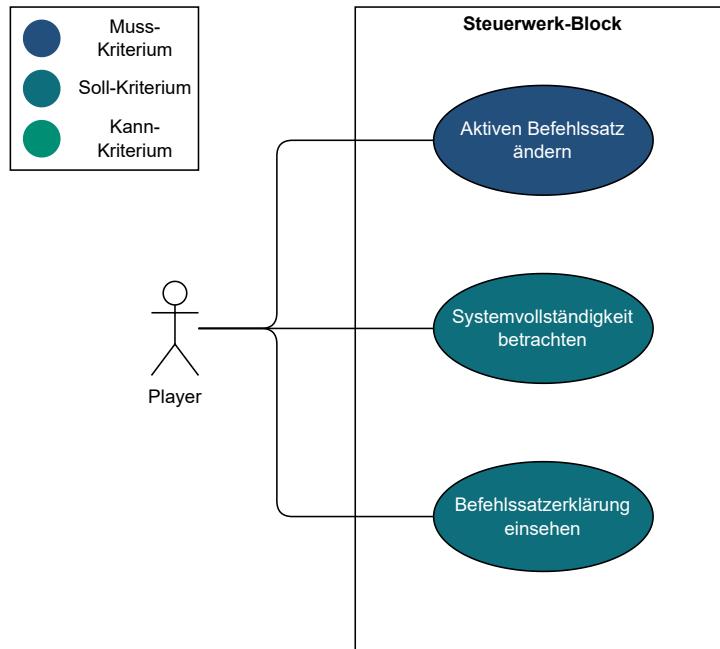


Abbildung 2.3: Use-Case-Diagramm: Steuerwerk-Block

Abbildung 2.3 zeigt die Anwendungsmöglichkeiten, die der Steuerwerk-Block bieten muss. Der Steuerwerk-Block soll dabei mit seiner GUI, die in Abbildung 5.9 dargestellt ist, Informationen zum Bau eines Lerncomputers anzeigen. Der Ablauf von Interaktionen mit dem Steuerwerk-Block wird im Aktivitätsdiagramm in Abbildung 2.8 dargestellt.

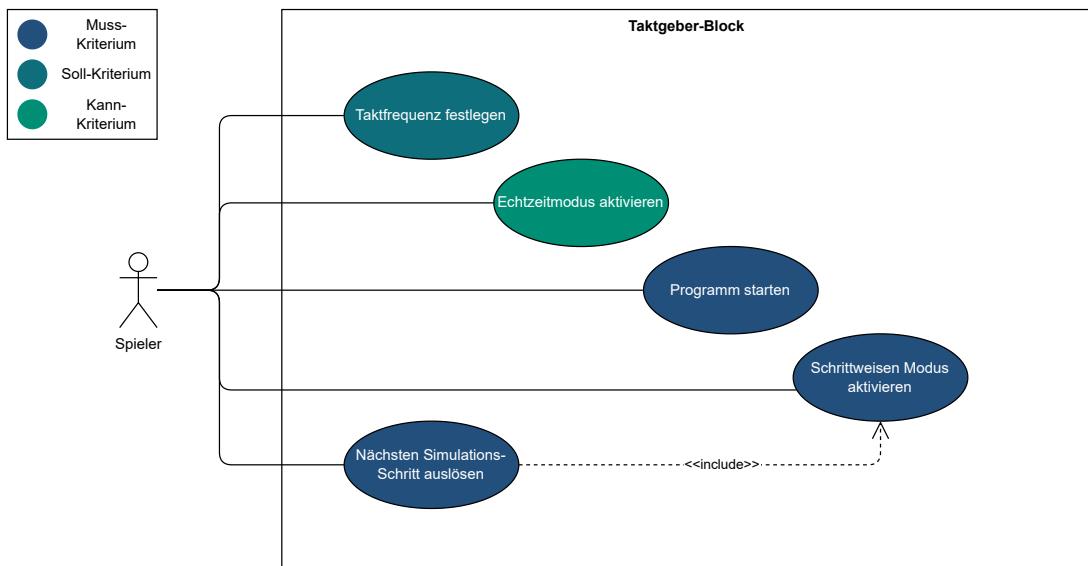


Abbildung 2.4: Use-Case-Diagramm: Taktgeber-Block

Abbildung 2.4 zeigt ein Use-Case-Diagramm für den Taktgeberblock. Der Akteur ist hier ein Spieler, der den Lerncomputer laufen lassen möchte. Die Use-Cases beschreiben die unterschiedlichen Modi, in welchen der Lerncomputer laufen kann.

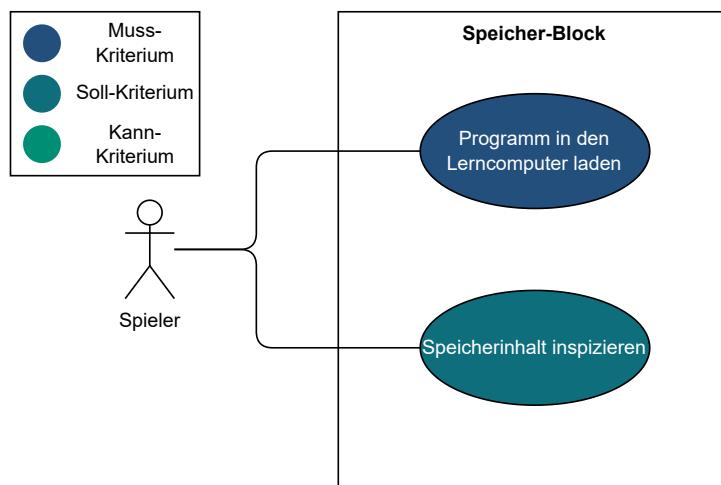


Abbildung 2.5: Use-Case-Diagramm: Speicher-Block

Abbildung 2.5 zeigt mögliche Interaktionen des Spielers mit dem Speicherblock. Besonders relevant für das Verständnis des Lerncomputers ist dabei der Einblick in den Speicher. Die Abläufe bei der Interaktion mit dem Speicher-Block werden im Aktivitätsdiagramm in Abbildung 2.10 genauer beschrieben.

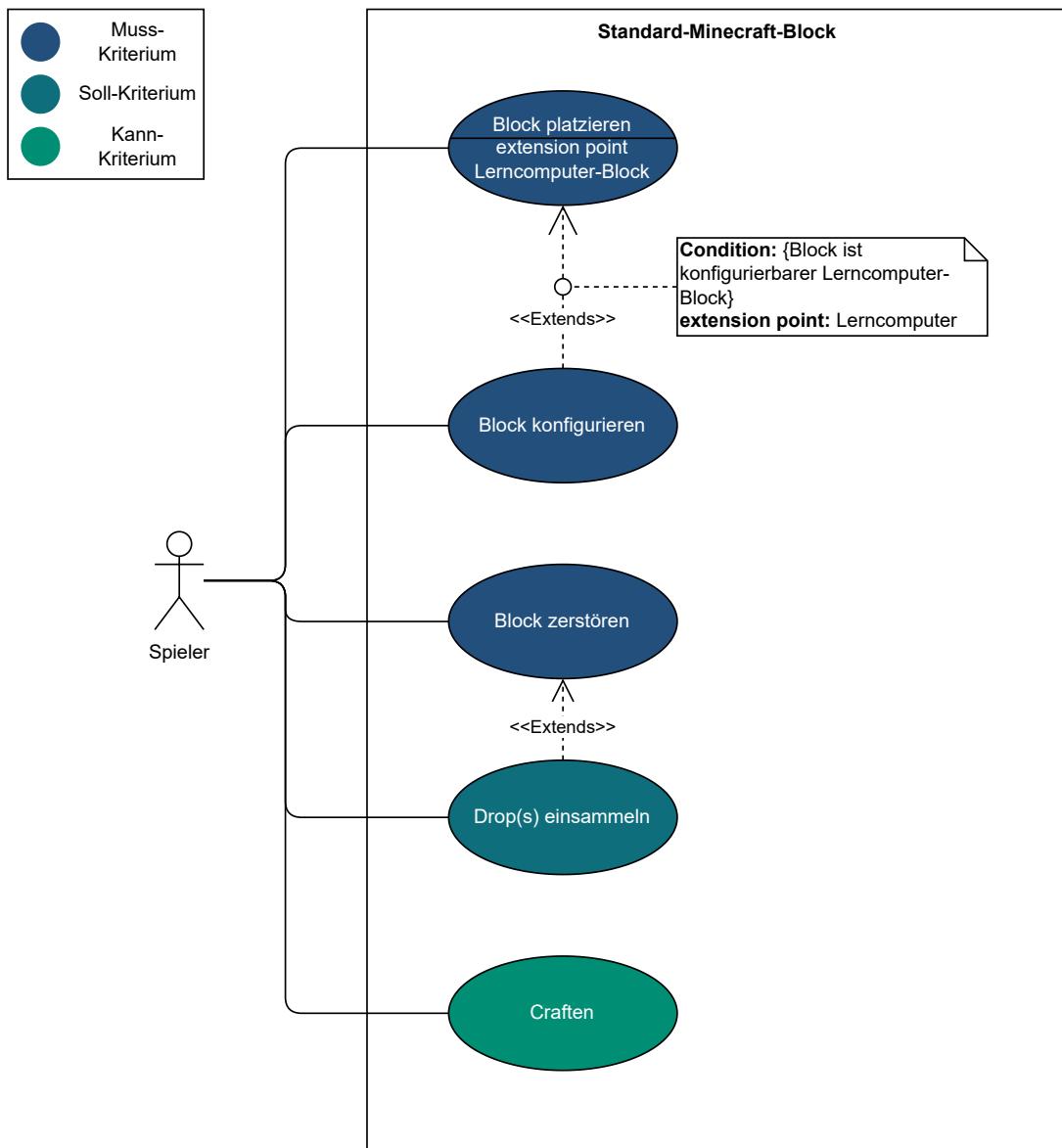


Abbildung 2.6: Use-Case-Diagramm: Standard-Minecraft-Block

Abbildung 2.6 zeigt die Interaktionen mit einem Standard-Minecraft-Block. Auch alle Blöcke der Mod sollen diese anbieten, um eine Einbindung in das Minecraft-Spiel zu schaffen.

## 2.3 Aktivitäten

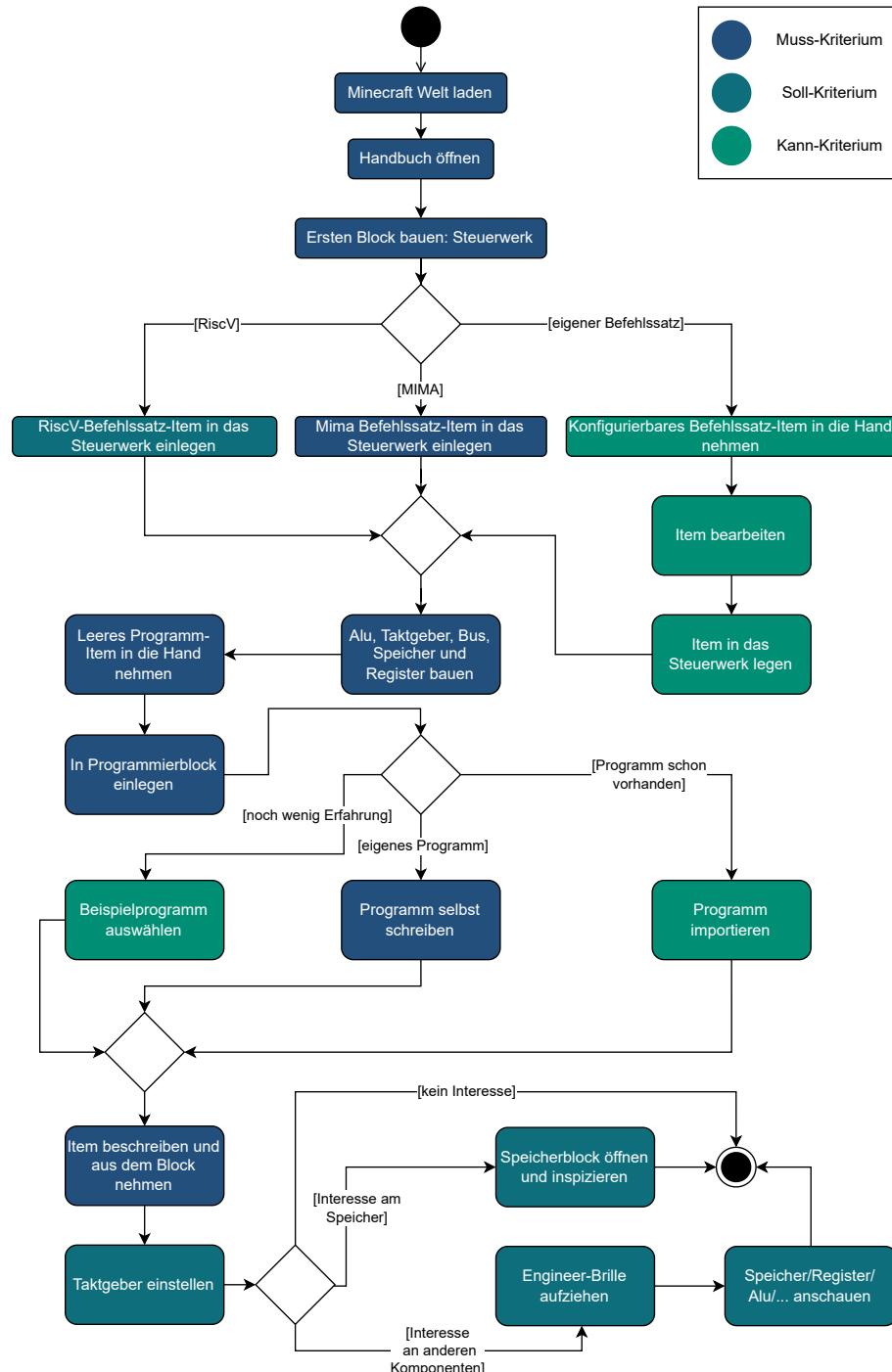


Abbildung 2.7: Aktivitätsdiagramm Gesamt

Abbildung 2.7 zeigt eine gesamte Übersicht über die Standardbenutzung des Lerncomputers und die verschiedenen Wahlmöglichkeiten des Spielers. Dabei wird der normale Durchlauf durch das Programm beschrieben.

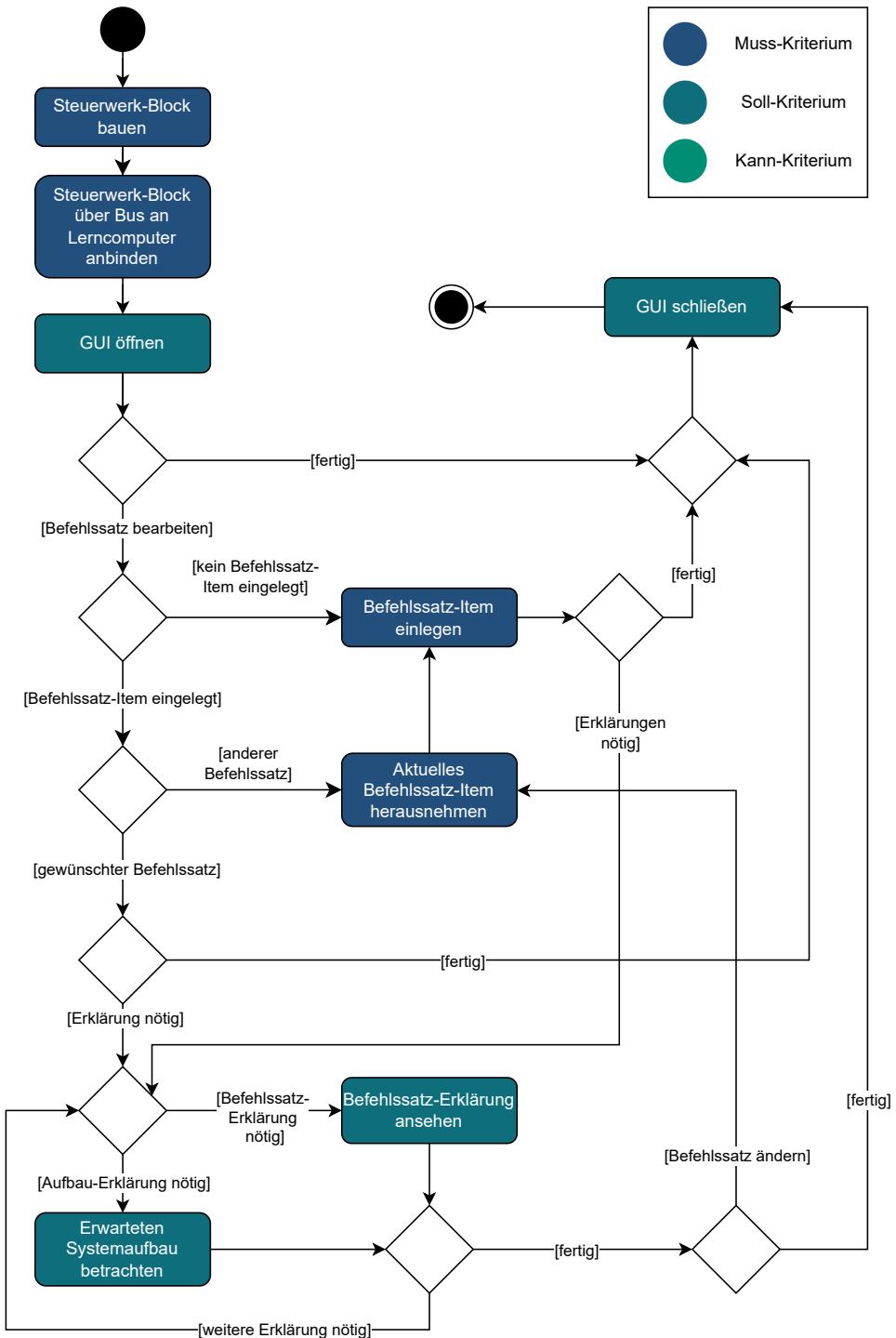


Abbildung 2.8: Aktivitätsdiagramm Steuerwerk

Abbildung 2.8 zeigt den Ablauf der Aktionen, die der Spieler in der GUI des Steuerwerk-Blocks, die in Abbildung 5.9 dargestellt ist, ausführen kann.

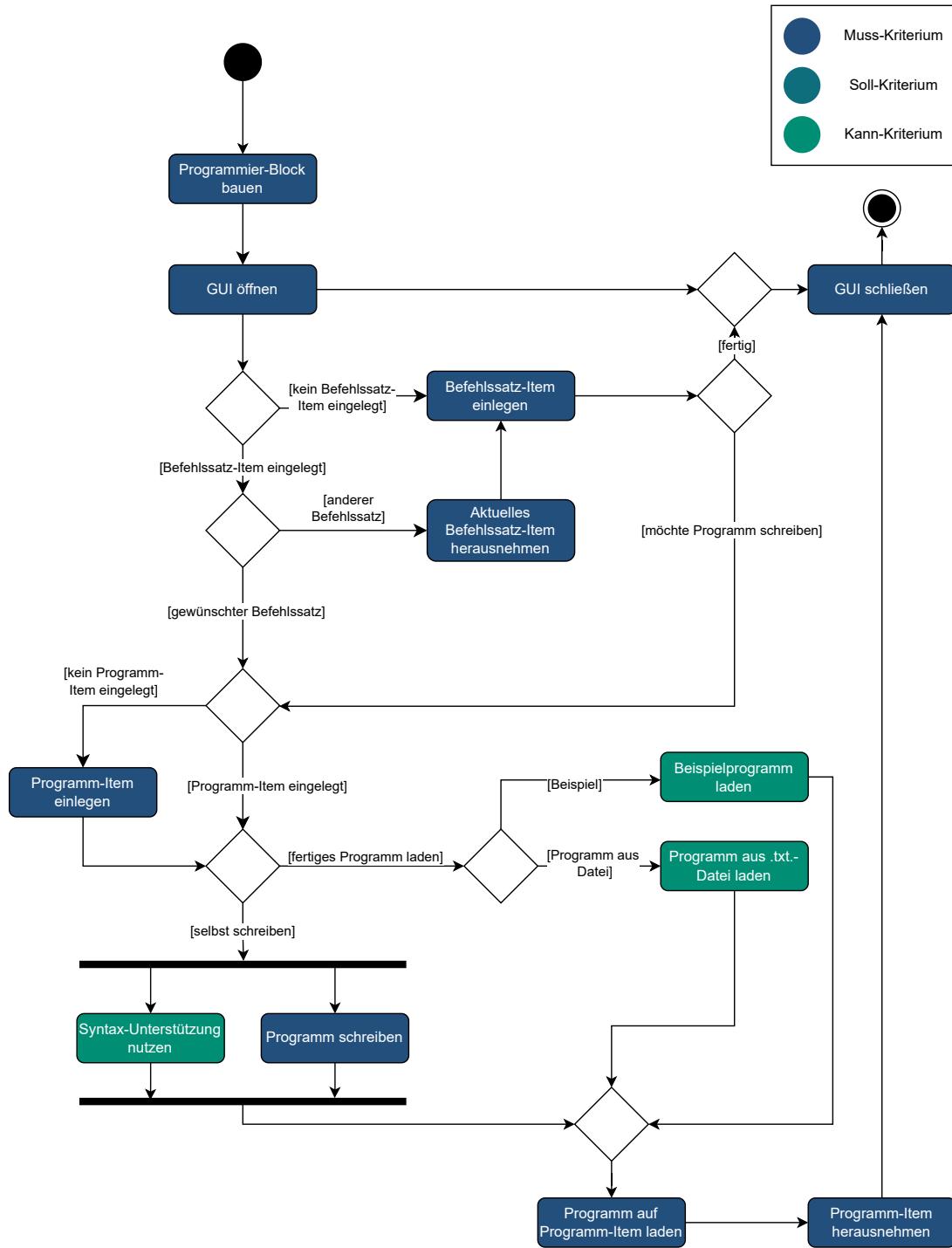


Abbildung 2.9: Aktivitätsdiagramm Programmier-Block

Abbildung 2.9 zeigt den Ablauf der Aktionen, die der Spieler in der GUI des Programmier-Blocks, die in Abbildung 5.5 dargestellt ist, ausführen kann.

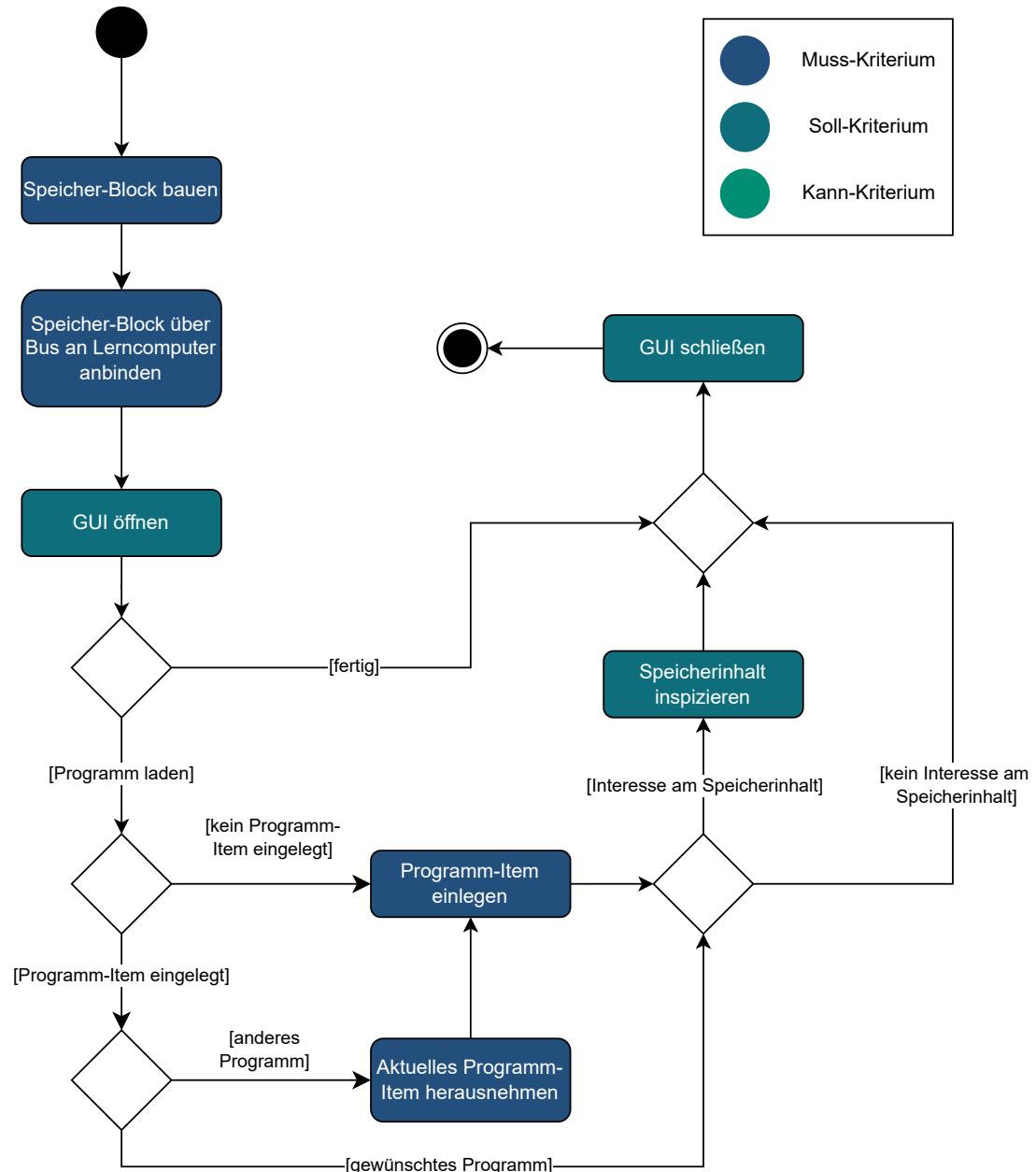


Abbildung 2.10: Aktivitätsdiagramm Speicher-Block

Abbildung 2.10 zeigt den Ablauf der Aktionen, die der Spieler in der GUI des Speicher-Blocks, die in Abbildung 5.4 dargestellt ist, ausführen kann.

## 3 Produktfunktionen

### 3.1 Welt-Funktionen

#### Lerncomputer-Erstellung $\langle F1 \rangle$

**Anwendungsfall:** Lerncomputer bauen (siehe Abbildung 2.1)

**Anforderung:**  $\langle RM1 \rangle \langle RM2 \rangle \langle RM3 \rangle \langle RM4 \rangle \langle RM5 \rangle \langle RS8 \rangle \langle RS9 \rangle \langle RS10 \rangle$

**Ziel:** Der Nutzer baut einen technisch korrekten Lerncomputer in der Minecraft-Welt.

**Vorbedingung:** Der Kreativmodus von Minecraft ist geöffnet. Es sind keine oder nur unverbundene Lerncomputer-Blöcke in der Minecraft-Welt vorhanden.

**Nachbedingung Erfolg:** Es ist ein technisch korrekter Lerncomputer nach einer angebotenen Architektur in der Minecraft-Welt vorhanden.

**Nachbedingung Fehlschlag:** Es ist ein technisch inkorrekt Lerncomputer, der keiner der angebotenen Architekturen entspricht, in der Minecraft-Welt vorhanden.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler entscheidet sich zum Bau eines Lerncomputers.

#### Beschreibung:

1. Steuerwerk bauen.
2. GUI des Steuerwerks öffnen.
3. Befehlssatz-Item einlegen, welches entweder
  - den MIMA-Befehlssatz und die zugehörige Architektur spezifiziert
  - oder
  - den RISC-V-Befehlssatz und die zugehörige Architektur spezifiziert.
4. Liste benötigter Bauteile betrachten.
5. Blöcke für alle benötigten Bauteile platzieren, insbesondere
  - ALU,
  - Taktgeber,
  - Hauptspeicher und
  - Register.
6. Alle Blöcke über eine Bus-Leitung aus Bus-Blöcken verbinden.
7. Programmier-Block platzieren.

**Erweiterung:**

- 3a Custom-Befehlssatz-Item wählen, GUI öffnen und den individuellen Befehlssatz sowie eine passende Architektur spezifizieren.
- 4a Grafische Visualisierung der Architektur betrachten.

**Lerncomputer-Entfernung**  $\langle F2 \rangle$

**Anwendungsfall:** Lerncomputer zerstören (siehe Abbildung 2.1)

**Anforderung:**  $\langle RM5 \rangle$

**Ziel:** Ein Lerncomputer Block wird aus der Minecraft Welt entfernt.

**Vorbedingung:** Der Spieler steht höchstens 4,5 Blöcke von dem Block entfernt und hat freie Sicht auf den Block.

**Nachbedingung Erfolg:** Das zum Block gehörende Item liegt als Drop auf dem Boden an der Stelle des Blockes.

**Nachbedingung Fehlschlag:** Der Block steht immer noch an der gleichen Stelle.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler hält seine linke Maustaste gedrückt, während er das Fadenkreuz auf den Block richtet.

**Beschreibung:** Man sieht die Standard-Minecraft-Abbauanimation, solange der Spieler auf den Block schaut. Macht er das eine block-spezifische Zeit, verschwindet der Block und der Drop erscheint.

**Lerncomputer-Konfiguration**  $\langle F3 \rangle$

**Anwendungsfall:** Lerncomputer konfigurieren (siehe Abbildung 2.1)

**Anforderung:**  $\langle RS1 \rangle \langle RS2 \rangle$

**Ziel:** Alle Blöcke des Lerncomputers sind konfiguriert, sodass der Lerncomputer starten kann.

**Vorbedingung:** Der Steuerwerk-Block besitzt ein Befehlssatz-Item und alle für die Architektur benötigten Blöcke.

**Nachbedingung Erfolg:** Alle Blöcke des Lerncomputers sind konfiguriert und der Lerncomputer kann gestartet werden.

**Nachbedingung Fehlschlag:** Nicht alle Blöcke des Lerncomputers sind konfiguriert und der Lerncomputer kann nicht gestartet werden.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler entscheidet sich dafür, den Lerncomputer zu konfigurieren.

**Beschreibung:**

1. Register konfigurieren.
2. Taktgeber konfigurieren.

## 3.2 Block-Funktionen

### Register-Konfiguration $\langle F4 \rangle$

**Anwendungsfall:** Register konfigurieren (siehe Abbildung 2.1)

**Anforderung:**  $\langle RS2 \rangle$

**Ziel:** Das Register erhält eine (neue) Funktion im Lerncomputer-Aufbau.

**Vorbedingung:** Spieler steht vor einem in der Minecraft-Welt platziertem Register-Block.

**Nachbedingung Erfolg:** Dem Register-Block ist eine Funktion im Lerncomputer zugewiesen.

**Nachbedingung Fehlschlag:** Der Register-Block hat keine Funktion im Lerncomputer.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler macht einen Rechtsklick auf den Register-Block.

**Beschreibung:**

1. GUI öffnen.
2. Register-Typ aus der Liste an Möglichkeiten auswählen.
3. GUI schließen.

### Taktgeber-Konfiguration $\langle F5 \rangle$

**Anwendungsfall:** Taktgeber konfigurieren (siehe Abbildung 2.1)

**Anforderung:**  $\langle RM11 \rangle \langle RS1 \rangle \langle RC9 \rangle$

**Ziel:** Der Taktmodus für die Berechnungen des Lerncomputers ist nach dem Wunsch des Spielers festgelegt.

**Vorbedingung:** Spieler steht vor einem in der Minecraft Welt platziertem Taktgeber-Block.

**Nachbedingung Erfolg:** Der Taktmodus für die Berechnungen des Lerncomputers ist eingesetzt.

**Nachbedingung Fehlschlag:** Der Taktgeber wird in den schrittweisen Modus gesetzt.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler macht einen Rechtsklick auf den Taktgeber-Block.

**Beschreibung:**

1. GUI öffnet sich.
2. Taktmodus auswählen aus
  - verschiedenen festen Taktfrequenzen,
  - schrittweiser Modus.
3. GUI schließen.

**Erweiterungen:**

- 1a Echtzeitmodus auswählen.

**Alternativen:**

- 1a Wenn kein Modus festgelegt wird, wird automatisch der schrittweise Modus eingestellt.

**Programmieren**  $\langle F6 \rangle$

**Anwendungsfall:** Programm schreiben (siehe Abbildung 2.2)

**Anforderung:**  $\langle RM4 \rangle \langle RM8 \rangle \langle RM9 \rangle \langle RC1 \rangle \langle RC13 \rangle$

**Ziel:** Ein fertiges Lerncomputer-Programm liegt auf einem Programm-Item vor.

**Vorbedingung:** Es liegen ein Programm-Item und ein Befehlssatz-Item im Programmiereditor-Block.

**Nachbedingung Erfolg:** Auf dem Programm-Item steht ein Programm, welches in Maschinencode des eingelegten Befehlssatzes übersetzt ist.

**Nachbedingung Fehlschlag:** Das Programm-Item ist unverändert.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler macht einen Rechtsklick auf den Programmier-Block.

**Beschreibung:**

1. GUI öffnet sich.
2. Cursor in den Texteditor setzen.
3. Zeile schreiben.
4. Zeilenumbruch einfügen.
5. Wiederholen ab 1, falls weitere Zeilen nötig.
6. Knopf drücken, um Programm fertigzustellen und auf das Programm-Item laden.
7. Programm-Item aus dem Programmieditor-Block nehmen.

**Erweiterung:**

- 1a Syntax-Unterstützung wird angezeigt.
- 1b Tipps werden in einem Nebenfenster angezeigt.

**Alternativen:**

- 4a Wenn das Programm nicht übersetzbare ist, schlägt die Fertigstellung fehl und es wird eine Fehlermeldung angezeigt.

**Programm laden**  $\langle F7 \rangle$

**Anwendungsfall:** Programm in Lerncomputer laden (siehe Abbildung 2.5)

**Anforderung:**  $\langle RM6 \rangle$

**Ziel:** Das Programm des eingelegten Items wird in den Speicher geladen.

**Vorbedingung:** Auf dem eingelegten Programm-Item ist ein valides Programm enthalten. Dies wird anhand des aktiven Befehlssatzes überprüft.

**Nachbedingung Erfolg:** Das Programm wurde erfolgreich in den Speicher geladen und kann jetzt gestartet werden.

**Nachbedingung Fehlschlag:** Das Programm konnte nicht geladen werden.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler hat ein beschriebenes Programm-Item in den Speicher gelegt.

**Beschreibung:**

1. Befehlssatz mit dem aktiven abgleichen.
2. Programm in den Speicher laden.
3. IAR mit Startwert beschreiben.
4. Neuen Speicherinhalt visualisieren.

**Alternativen:**

- 1a Wenn der Befehlssatz nicht übereinstimmt, wird eine Fehlermeldung angezeigt.

**Speicher inspizieren**  $\langle F8 \rangle$

**Anwendungsfall:** Speicherinhalt inspizieren (siehe Abbildung 2.5)

**Anforderung:**  $\langle RS4 \rangle$

**Ziel:** Der Spieler kann den aktuellen Speicherinhalt betrachten.

**Vorbedingung:** Der Speicher-Block ist platziert.

**Nachbedingung Erfolg:** Der Spieler ist über den aktuellen Speicherinhalt informiert.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler macht einen Rechtsklick auf den Speicher-Block.

**Beschreibung:**

1. Die GUI öffnet sich.
2. Einige Speicherstellen werden mit ihrem Inhalt in Hexadezimal-Darstellung angezeigt.
3. Der Spieler kann durch Scrollen Speicherstellen betrachten.
4. Der Spieler schließt die GUI.

**Taktgeber-Block  $\langle F9 \rangle$**

**Anwendungsfall:** Programm starten (siehe Abbildung 2.4)

**Anforderung:**  $\langle RM11 \rangle$

**Ziel:** Das Programm wird ausgeführt.

**Vorbedingung:** Ein valides Programm wurde in den Speicher-Block geladen.

**Nachbedingung Erfolg:** Das Programm wurde ausgeführt.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler hat einen Takt mit dem Taktgeber-Block ausgelöst.

**Beschreibung:**

1. Es werden, nach der jeweiligen Befehlsarchitektur, Befehle geladen.
2. Diese Befehle werden ausgeführt.

**Befehlssatz  $\langle F10 \rangle$**

**Anwendungsfall:** Aktiven Befehlssatz ändern (siehe Abbildung 2.3)

**Anforderung:**  $\langle RM7 \rangle$

**Ziel:** Der Lerncomputer wird auf einen anderen Befehlssatz, möglicherweise eine andere Rechnerarchitektur, umgestellt.

**Vorbedingung:** Es muss bereits ein Steuerwerk-Block und ein Befehlssatz-Item geben.

**Nachbedingung Erfolg:** Der neue Befehlssatz wurde erfolgreich geladen und der Lerncomputer auf diesen umgestellt.

**Nachbedingung Fehlschlag:** Der Befehlssatz konnte nicht geladen werden, daher ist kein Befehlssatz festgelegt.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler legt ein (neues) Befehlssatz-Item in den Steuerwerk-Block.

**Beschreibung:**

1. Befehlssatz auf Validität überprüfen.
2. Bedingungen des Befehlssatzes laden und dem Spieler Diskrepanzen anzeigen.
3. Befehlssatz umstellen.

**Alternativen:**

- 1a Wenn der Befehlssatz nicht valide ist, das Laden abbrechen.

**Steuerwerk**  $\langle F11 \rangle$

**Anwendungsfall:** Systemvollständigkeit betrachten (siehe Abbildung 2.3)

**Anforderung:**  $\langle RS6 \rangle$

**Ziel:** Der Spieler erhält Kenntnis über die Funktionsfähigkeit des Lerncomputers, bzw. die Schritte, die benötigt sind, um diese zu erreichen.

**Vorbedingung:** Es existiert ein Steuerwerk, mit dem der Spieler interagiert.

**Nachbedingung Erfolg:** Der Spieler hat eine Hilfe beim Bau des Lerncomputers erhalten.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Spieler nutzt das GUI des Steuerwerk-Blocks.

**Beschreibung:**

1. Überprüfen, welche Bestandteile für den Befehlssatz benötigt werden, als auch welche davon bereits vorhanden sind.
2. Diskrepanzen werden dem Spieler mitgeteilt.

### 3.3 Item-Funktionen

**Handbuch**  $\langle F12 \rangle$

**Anwendungsfall:** Mod-Grundlagen nachlesen (siehe Abbildung 2.1)

**Anforderung:**  $\langle RM10 \rangle$

**Ziel:** Der Spieler kann nachlesen, wie die Lerncomputer-Blöcke benutzt werden können.

**Vorbedingung:** Der Spieler hat das Handbuch-Item in der Schnellzugriffsleiste und ausgewählt in der Hand.

**Nachbedingung Erfolg:** Der Spieler ist informiert und schließt die GUI wieder.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler hält das Handbuch-Item in der Hand und der Benutzer macht einen Rechtsklick mit der Maus.

**Beschreibung:** Das Handbuch öffnet sich. Es besitzt mehrere Seiten, durch die der Spieler mithilfe von GUI-Elementen blättern kann. Die Nutzeroberfläche ist in Abbildung 5.11 dargestellt. Es wird beschrieben:

- Wie man eine MIMA mit den Lerncomputermod-Blöcken bauen kann.
- Wie man ein MIMA-Programm schreibt.
- Wie man das Programm-Item beschreibt und es in den Speicher einlegen kann.
- Wie man den Lerncomputer startet.
- Wie man beobachten kann, was der Lerncomputer beim Laufen macht.
- Wie man den Befehlssatz ändert.
- Die textuellen Erklärungen werden durch Grafiken unterstützt.
- Es wird rudimentär beschrieben, wie das Befehlssatz-Item konfiguriert werden kann.

**Erweiterung:**

1a, 2a Die Beschreibungen existieren auch für den RISC-V-Befehlssatz.

**Brillen-Item** *(F13)*

**Anwendungsfall:** Lerncomputer-Status beobachten (siehe Abbildung 2.1)

**Anforderung:** *(RS5)*

**Ziel:** Der Spieler kann von außen Informationen über den inneren Zustand von Lerncomputer Blöcken erhalten.

**Vorbedingung:** Der Spieler hat das Brillen-Item in den Helm Slot in seinem Inventar gelegt.

**Nachbedingung Erfolg:** In der Spielansicht ist oben das Brillen-Overlay, welches in Abbildung 5.2 dargestellt ist, zu sehen.

**Akteure:** Der Spieler

**Auslösendes Ereignis:** Der Spieler richtet das Fadenkreuz auf einen Lerncomputer-Block.

**Beschreibung:**

- Schaut der Spieler auf eine ALU, sieht er Modus, Ausgabewert und die beiden Eingabewerte angezeigt.
- Schaut der Spieler auf ein Register, sieht er den aktuellen Wert.
- Schaut der Spieler auf einen Bus-Block, sieht er den Wert, der gerade auf ihm liegt.

- Schaut der Spieler auf einen I/O-Block, sieht er seinen aktuellen Wert
- Schaut der Spieler auf den Taktgeber, sieht er den aktuellen Takt.
- Schaut der Spieler auf das Steuerwerk, sieht er den aktuellen Befehlssatz.

**Erweiterung:**

Die textuelle Ansicht wird durch kleine, an die Blöcke angepasste Grafiken unterstützt.

## 4 Nichtfunktionale Anforderungen

### 4.1 Funktionalität

Produktqualität	sehr gut	gut	normal	nicht relevant
Angemessenheit				x
Richtigkeit	x			
Interoperabilität			x	
Ordnungsmäßigkeit				x

Es wird keine Prüfung auf Angemessenheit der Inhalte durchgeführt, da durch diese Modifikation keine unangemessenen Elemente eingeführt werden. Die Richtigkeit von Berechnungen unter Nutzung der Modifikation muss zu jeder Zeit sichergestellt sein. Zudem müssen Interaktionen mit dem Minecraft-Basiscode zu jeder Zeit funktionieren. Direkte Interaktionen mit weiteren Modifikationen werden nicht unterstützt. Die Ordnungsmäßigkeit des Projekts wird nicht geprüft, da keine rechtlich relevanten Inhalte erstellt werden.

### 4.2 Sicherheit

Produktqualität	sehr gut	gut	normal	nicht relevant
Zuverlässigkeit		x		
Reife	x			
Fehlertoleranz			x	
Wiederherstellbarkeit			x	

Minecraft bietet keine vollständige Zuverlässigkeit. Die Modifikation darf keine weiteren Schwachstellen eröffnen, jedoch können Minecraft-seitige Unsicherheiten nicht behoben werden. Das Produkt soll bei Fehlerzuständen, die direkt aus der Nutzung der Modifikation resultieren, nicht versagen. Die Modifikation soll Anwendungsfehler jedoch nicht korrigieren. Bei Versagen können Daten und Zustände nicht über durch Minecraft gespeicherte Inhalte hinaus gesichert werden.

## 4.3 Benutzbarkeit

Produktqualität	sehr gut	gut	normal	nicht relevant
Verständlichkeit	x			
Erlernbarkeit	x			
Bedienbarkeit		x		
Effizienz				x
Zeitverhalten		x		
Verbrauchsverhalten				x

Das Produkt ist eine Lernanwendung, weshalb hohe Verständlichkeit und Erlernbarkeit im Rahmen der Vorkenntnisse des Nutzers gegeben sind. Minecraft-Erfahrung wird für die Bedienung vor Vorteil sein, da die Modifikation auf Minecraft-Konzepte zurückgreift. Als Lerncomputer wird das Produkt nicht auf Effizienz ausgerichtet. Das Produkt bietet nichtsdestotrotz eine korrekte Repräsentation des vom Nutzer eingestellten Zeitverhaltens. Das Verbrauchsverhalten der Modifikation wird nicht optimiert.

## 4.4 Änderbarkeit

Produktqualität	sehr gut	gut	normal	nicht relevant
Analysierbarkeit	x			
Modifizierbarkeit	x			
Stabilität			x	
Prüfbarkeit			x	
Übertragbarkeit				x
Anpassbarkeit				x
Installierbarkeit	x			
Konformität				x
Austauschbarkeit				x

Das Produkt wird umfangreich dokumentiert und erweiterbar entwickelt. Änderungen können allerdings nur eingeschränkt automatisiert getestet werden, für vollständige Prüfung sind umfangreiche Nutzertests nötig. Auf Übertragbarkeit der Inhalte in anderen Softwareumgebungen sowie dafür nötige Anpassungen wird nicht geachtet. Die Modifikation ist leicht direkt über den Fabric-Mod-Loader zu installieren.

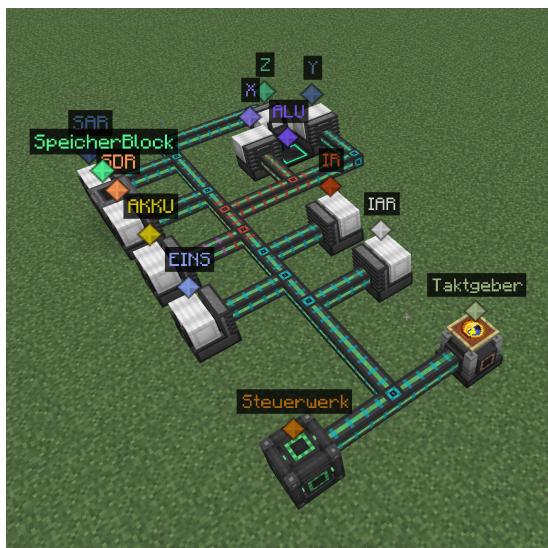
## 4.5 Nichtfunktionale Produktanforderungen

- ⟨Q1⟩ Die Mod stellt die fehlerfreie Ausführung inhaltlich korrekter Programme basierend auf einem korrekten Befehlssatz sicher.
- ⟨Q2⟩ Die Mod arbeitet ohne für den Nutzer erkennbare Umwege und Schwierigkeiten mit Minecraft zusammen.
- ⟨Q3⟩ Die Modifikation erhöht die Versagenshäufigkeit von Minecraft nicht.
- ⟨Q4⟩ Die Mod ist intuitiv und leicht verständlich aufgebaut und insbesondere für Nutzer mit Minecraft-Vorkenntnissen sehr leicht zu erlernen und zu bedienen.
- ⟨Q6⟩ Die Mod ist modular aufgebaut, leicht erweiterbar und verständlich dokumentiert.
- ⟨Q7⟩ Die Ausführung jedes möglichen Simulationstaktes des Lerncomputers benötigt eine Maximaldauer von einem Minecraft-Tick.
- ⟨Q8⟩ Die GUIs und Datenausgaben werden auf Deutsch und Englisch angeboten.
- ⟨Q9⟩ Der Projektcode wird unter einer MIT-Lizenz auf GitHub veröffentlicht.
- ⟨Q10⟩ Die Mod ist als Fabric-Mod exportierbar.
- ⟨Q11⟩ Die Mod wird auf Curse-Forge veröffentlicht.

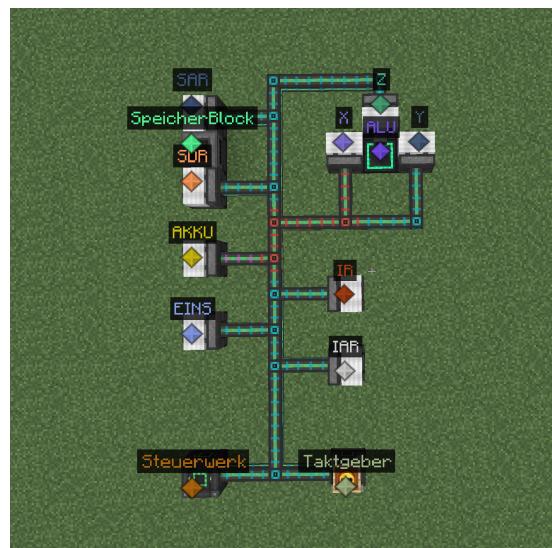
## 5 Benutzeroberfläche

### 5.1 Lerncomputer in der Minecraft-Welt

Ein fertig gebauter Lerncomputer in einer Minecraft Welt soll alle architektur-relevanten Bauteile beinhalten, wie in Abbildung 5.1 zu sehen. Die Markierungen in den Ansichten dienen nur der Übersicht und werden nicht umgesetzt.



(a) Ansicht von der Seite



(b) Ansicht von Oben

Abbildung 5.1: Ansicht des MIMA-Computeraufbaus in Minecraft

## 5.2 Brillen-Item

Richtet der Spieler sein Fadenkreuz auf einen Lerncomputer-Block und trägt die Brille, so erscheinen Informationen. Abbildung 5.2 zeigt das exemplarisch für den Akku einer MIMA.



Abbildung 5.2: GUI des Brillenitems

### 5.3 Register-Block

In der GUI des Register-Blocks kann man sehen, welche Register bereits am Bus angeschlossen sind, und dann einen Typ für das aktuelle Register auswählen. Der GUI-Aufbau ist in Abbildung 5.3 zu sehen.

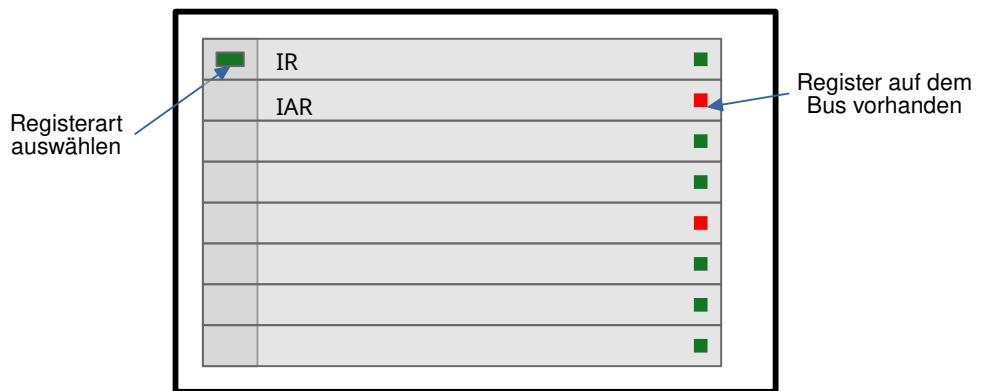


Abbildung 5.3: GUI des Register-Blocks

## 5.4 Speicher-Block

In der GUI des Speicher-Blocks kann der Spieler ein Programm laden und den aktuellen Speicherinhalt inspizieren, wie in Abbildung 5.4 dargestellt.

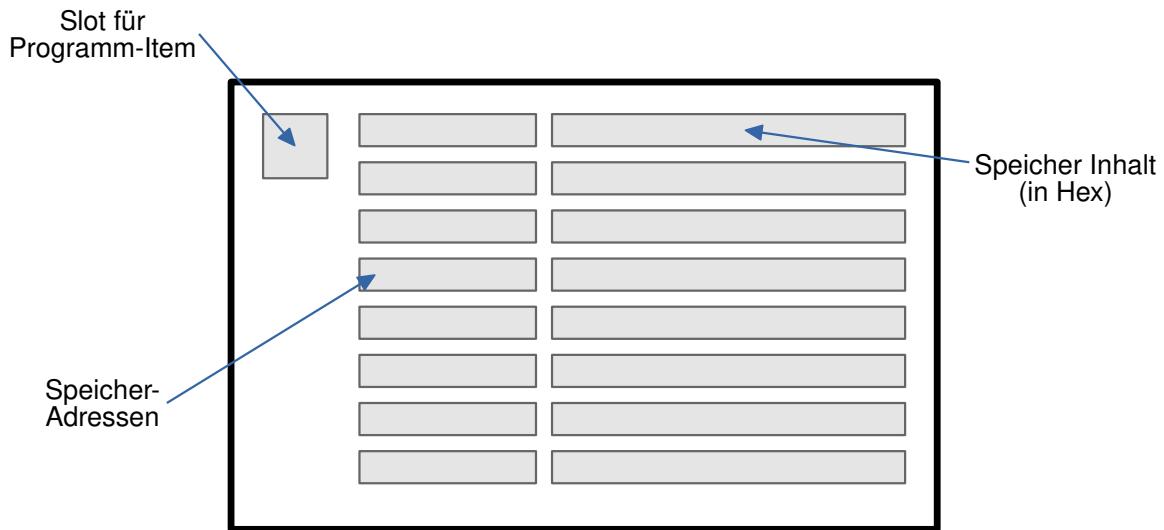


Abbildung 5.4: GUI des Speicherblocks

## 5.5 Programmier-Block

In der GUI des Programmier-Blocks kann der Spieler ein Programm schreiben und anschließend auf ein Programm-Item laden, wie in Abbildung 5.5 dargestellt. Abbildung 5.6 zeigt die Darstellung von Informationen zu den Befehlen im aktuellen Befehlssatz bei Nutzung des Hilfe-Knopfes.

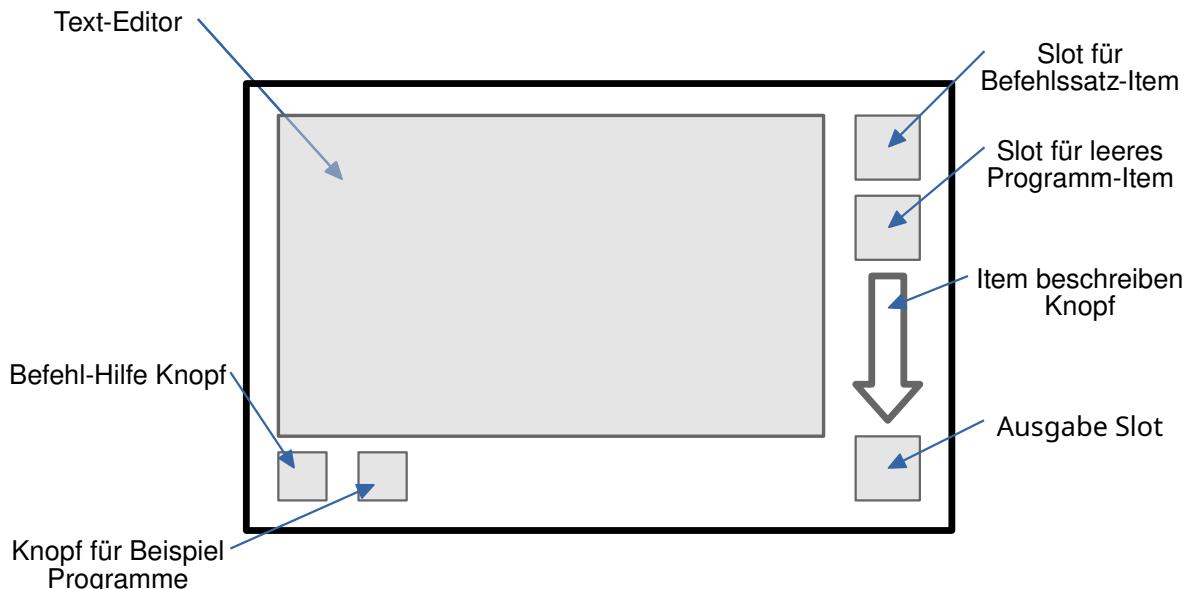


Abbildung 5.5: GUI des Programmier-Blocks

OpCode	Mnemonik	Beschreibung
0	LDC	c -> Akku
1	LDV	a <a> -> Akku
2	STV	Akku -> <a>
3	ADD	Akku + <a> -> Akku
4	AND	Akku AND <a> -> Akku
5	OR	Akku OR <a> -> Akku
6	XOR	Akku XOR <a> -> Akku
7	EQL	a falls Akku = <a>:-1 -> Akku sonst : 0 -> Akku
8	JMP	a -> IAR
9	JMN	falls Akku < 0 : a -> IAR
F0	HALT	stoppt die MIMA
F1	NOT	bilde Eins-Komplement von Akku -> Akku
F2	RAR	rotiere Akku eins nach rechts -> Akku

Abbildung 5.6: GUI des Programmier-Blocks mit Hilfenfenster

## 5.6 Terminal-Block

Der Terminal-Block dient zur Aus- und Eingabe von Texten und ahmt eine Kommandozeile nach, wie in Abbildung 5.7 zu sehen.

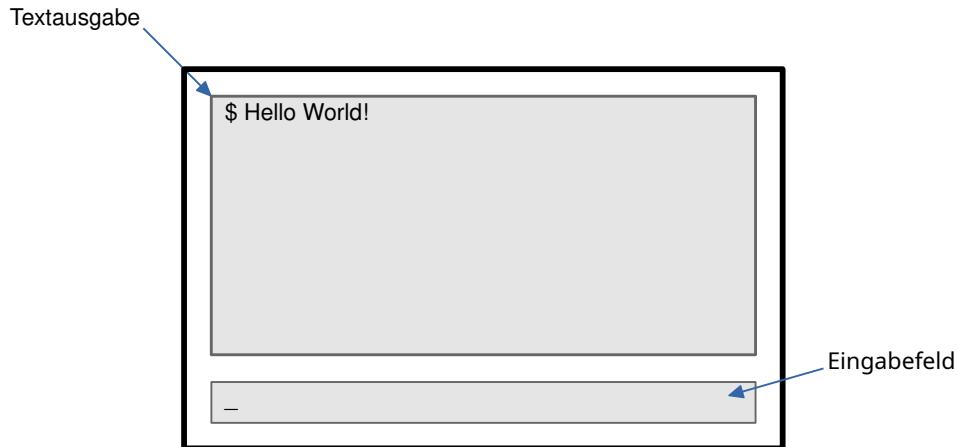


Abbildung 5.7: GUI des Terminal-Blocks

## 5.7 Konfigurierbare Befehlssatz-Item

In der GUI des konfigurierbaren Befehlssatz-Items, welche in Abbildung 5.8 dargestellt ist, kann der Spieler die Spezifikation für einen eigenen Befehlssatz für den Lerncomputer festlegen. Beim Speichern der Spezifikation im .json-Format findet keine Semantik-Überprüfung statt.

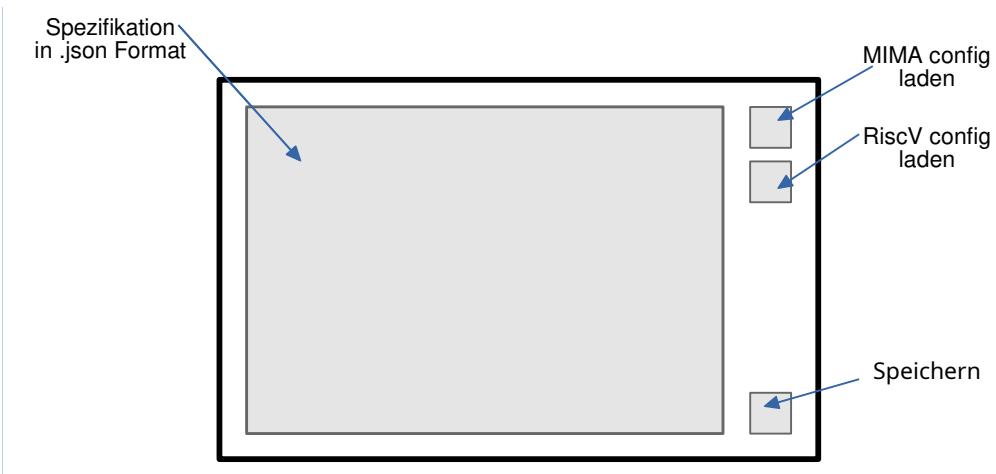


Abbildung 5.8: GUI des konfigurierbaren Befehlssatz-Items

## 5.8 Steuerwerk-Block

In der GUI des Steuerwerks, wie in Abbildung 5.9 gezeigt, wird ein Befehlssatz-Item in den Lerncomputer eingelegt. Auf der linken Seite ist der Aufbau für den eingelegten Befehlssatz gezeigt. Bei einem selbst konfiguriertem Befehlssatz-Item steht hier nur die .json-Spezifikation. Rechts sind alle für die Architektur benötigten Komponenten gelistet und es ist angezeigt, ob diese bereits angebunden sind.

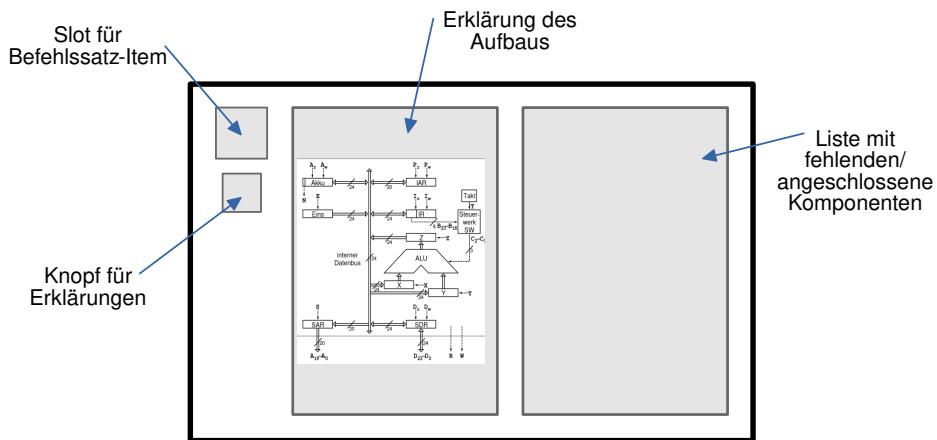


Abbildung 5.9: GUI des Steuerwerk-Blocks

## 5.9 Taktgeber-Block

In der GUI des Taktgeber-Blocks lässt sich die Taktfrequenz und damit der Taktmodus des Lerncomputers einstellen. Der Schieber, welcher in Abbildung 5.10 zu sehen ist, ist nicht stufenlos verstellbar, sondern hat Abstufungen.

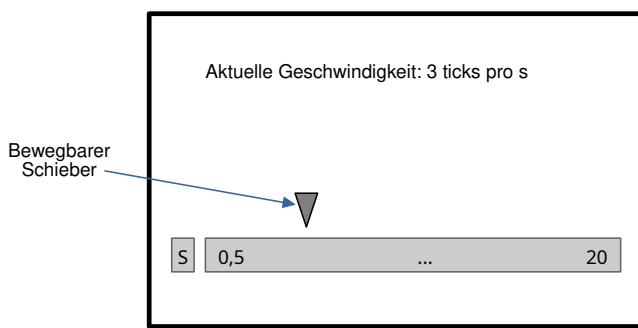


Abbildung 5.10: GUI des Taktgeber-Blocks

## 5.10 Handbuch

Das Handbuch bietet in seiner GUI, wie in Abbildung 5.11 zu sehen, die Möglichkeit, durch die Mod-Dokumentation zu blättern.

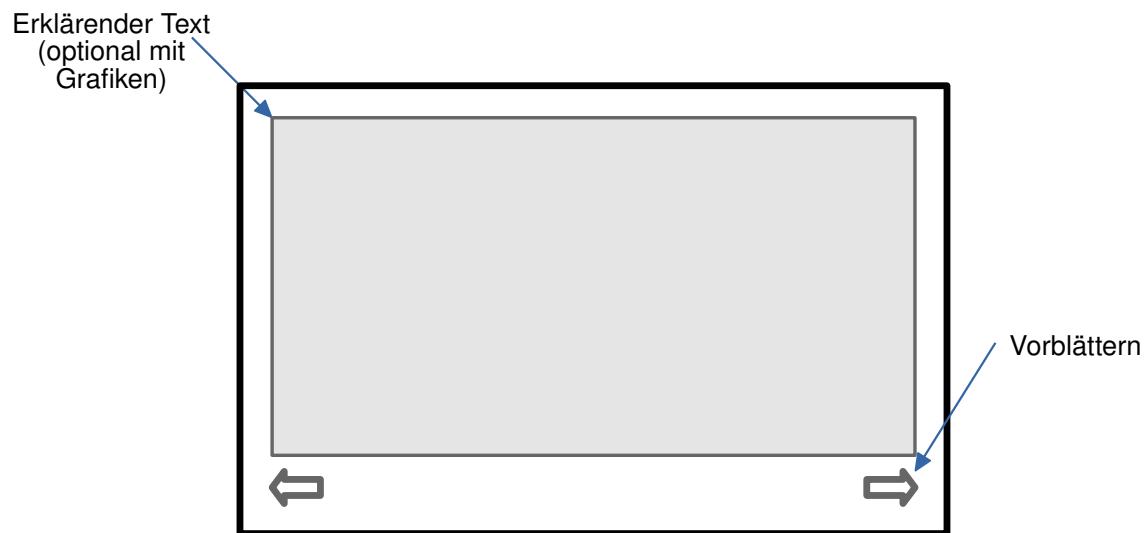


Abbildung 5.11: GUI des Handbuchs

## 6 Technische Produktumgebung

### 6.1 Software

Server-Betriebssystem: Minecraft-Mindestanforderungen (Windows 7 und höher ODER macOS 10.14.5 Mojave und höher ODER aktuelles Linux/BSD)

Client-Betriebssystem: Minecraft-Mindestanforderungen (Windows 7 und höher ODER macOS 10.14.5 Mojave und höher ODER aktuelles Linux/BSD)

Weitere Programme: Minecraft-Java-Version 1.20.2, Fabric-Mod-Loader Version 0.14.24

### 6.2 Hardware

Server: Minecraft-Mindestanforderungen (x64-System, mindestens 2 GB RAM, Intel Core i3-3210 3,2 GHz/AMD A8-7600 APU 3,1 GHz/Apple M1 oder ähnlich)

Client: Minecraft-Mindestanforderungen (x64-System, mindestens 2 GB RAM, Intel Core i3-3210 3,2 GHz/AMD A8-7600 APU 3,1 GHz/Apple M1 oder ähnlicher Prozessor, Intel HD Graphics 4000 | AMD Radeon R5 oder ähnlich)

## 7 Glossar

**Assembler:** Ein Assembler ist ein Computerprogramm, das Assemblersprachcode in Maschinencode übersetzt. Assemblersprache ist eine Low-Level-Programmiersprache, die eng mit dem Maschinencode verwandt ist, aber für Menschen leichter zu lesen und zu schreiben ist.<sup>1</sup>

**Befehlssatz:**

Der Befehlssatz eines Prozessors ist in der Rechnerarchitektur die Menge der Maschinenbefehle, die ein bestimmter Prozessor ausführen kann. Je nach Prozessor variiert der Umfang des Befehlssatzes zwischen beispielsweise 33 und über 500 Befehlen. CISC-Prozessoren haben tendenziell größere Befehlssätze als RISC-Prozessoren.<sup>2</sup>

**Block:**

Ein Block ist ein besonderer Gegenstand, den man in der Minecraft-Welt platzieren kann. Nahezu die gesamte Spielwelt besteht aus Blöcken.<sup>3</sup>

**Drop:**

Der Begriff Drop bezeichnet fallen gelassene (vom engl. dropped) Objekte, die in der Spielwelt auf dem Boden schweben und aufgenommen werden können. Als Drop sind sie keine Gegenstände, sondern Objekte, denn sie bewegen sich.<sup>4</sup>

**Kreativmodus:**

Einer der 5 Modi in Minecraft. Wie der Name schon andeutet, ist der Kreativmodus (engl. Creativemode) speziell für das kunstvolle Erschaffen von umfangreichen, komplexen Bauwerken konzipiert. Dafür bietet er dem Spieler unendlich viele Ressourcen und Unsterblichkeit.<sup>5</sup>

**Inventar:**

Im Überlebensmodus ist das Inventar der Speicher des Spielers für Blöcke und sonstige Gegenstände, sozusagen ein Rucksack, den man immer bei sich trägt. Das Inventar hat ein begrenztes Fassungsvermögen. Im Kreativmodus ist das Inventar kein Speicher, sondern eine Auswahl fast aller Blöcke und sonstiger Gegenstände in unbegrenzter Menge.<sup>6</sup>

---

<sup>1</sup>Quelle: <https://techwatch.de/blog/understanding-the-basics-what-is-an-assembler-and-how-does-it-work/>

<sup>2</sup>Quelle: <https://de.wikipedia.org/wiki/Befehlssatz>

<sup>3</sup>Vgl.: <https://minecraft.fandom.com/de/wiki/Block>

<sup>4</sup>Quelle: <https://minecraft.fandom.com/de/wiki/Drop>

<sup>5</sup>Vgl.: <https://minecraft.fandom.com/de/wiki/Kreativmodus>

<sup>6</sup>Quelle: <https://minecraft.fandom.com/de/wiki/Inventar>

**Item:**

Ein Gegenstand (engl. Item) ist alles, was man in sein Inventar aufnehmen und in der Hand halten kann. Ein Block ist ein besonderer Gegenstand, den man in der Welt platzieren kann (z.B. Erde oder eine Werkbank). Daneben gibt es noch einige Gegenstände, die beim Platzieren zu einem beweglichen Objekt werden, z.B. ein Boot.<sup>7</sup>

**MIMA:**

Die mikroprogrammierte Minimalmaschine (MIMA) ist ein Lehrmodell zur vereinfachten Darstellung von Mikroprozessoren, basierend auf der Von-Neumann-Architektur, welche von Tamim Asfour am Karlsruher Institut für Technologie entwickelt wurde.<sup>8</sup>

**Minecraft-Objekte:**

Objekte (engl. Entities) sind neben den Gegenständen, zu denen auch die Blöcke gehören, die andere große Gruppe der Spielelemente in Minecraft. Eine Minecraft-Welt besteht aus Blöcken und Objekten. Im Gegensatz zu den Blöcken sind die Objekte meist beweglich (z.B. eine Lore oder Projektil). Zu den Objekten zählen auch jegliche Drops.<sup>9</sup>

**Mod:**

Als Modifikation (Abkürzung Mod) wird alles bezeichnet, das den Spielinhalt von Minecraft verändert.<sup>10</sup>

**Redstone:**

Redstone ist ein flacher, transparenter Block, der Redstone-Signale übertragen kann. Ein Signal geht von einem Signalblock über den Redstone zu einem Empfängerblock und löst dort eine Aktion aus.

**Rezepte:**

Rezepte geben an, mit welchen Gegenständen ein Block oder Gegenstand in Minecraft hergestellt werden kann.

**RISC-V:**

RISC-V ist eine Befehlssatzarchitektur, die sich auf das Designprinzip des Reduced Instruction Set Computers (RISC) stützt. Das Designziel von RISC ist der Verzicht auf einen komplexen Befehlssatz hin zu einfach zu dekodierenden und schnell auszuführenden Befehlen.<sup>11</sup>

---

<sup>7</sup>Quelle: <https://minecraft.fandom.com/de/wiki/Gegenstand>

<sup>8</sup>Vgl.: [https://de.wikipedia.org/wiki/Mikroprogrammierte\\_Minimalmaschine](https://de.wikipedia.org/wiki/Mikroprogrammierte_Minimalmaschine)

<sup>9</sup>Vgl.: <https://minecraft.fandom.com/de/wiki/Objekt>

<sup>10</sup>Quelle: <https://minecraft.fandom.com/de/wiki/Modifikation>

<sup>11</sup>vgl.: <https://de.wikipedia.org/wiki/RISC-V> und [https://de.wikipedia.org/wiki/Reduced\\_Instruction\\_Set\\_Computer](https://de.wikipedia.org/wiki/Reduced_Instruction_Set_Computer)

**Schnellzugriffsleiste (engl. Hotbar) :**

Neun Slots des Inventars bilden die Schnellzugriffsleiste, die stets am unteren Fensterrand angezeigt wird. Einer der Slots der Schnellzugriffsleiste ist der aktuelle Slot. Den Inhalt dieses Slots hält man in der Hand. <sup>12</sup>

**Slot:**

Ein Slot in Minecraft ist ein Platz in der GUI, an den Items gelegt werden können. Das Inventar und Kisten bestehen aus Slots.

**Überlebensmodus:**

Einer der 5 Modi in Minecraft. Der Schwerpunkt des Spiels liegt beim Überlebensmodus (engl. Survivalmode), wie der Name schon sagt, auf dem Überleben des Spielers. <sup>13</sup>

---

<sup>12</sup>Quelle: [https://minecraft.fandom.com/de/wiki/Inventar\#Inventar\\_im\\_\u00c3\u0099berlebensmodus](https://minecraft.fandom.com/de/wiki/Inventar\#Inventar_im_\u00c3\u0099berlebensmodus)

<sup>13</sup>Vgl.: [https://minecraft.fandom.com/de/wiki/\"](https://minecraft.fandom.com/de/wiki/\)Überlebensmodus