Der Tic-Tac-Toe-Robo

Karsten Schulz, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In den letzten Jahren etablierten sich viele Spiele, die man gegen einen Computer spielen kann. Der hier entwickelte Tic-Tac-Toe-Robo stellt eine weitere Möglichkeit dar. Unter der Verwendung von LEGO® zur Konstruktion und LEGO® Mindstorms® EV3 zur Ansteuerung wird eine weitere Umsetzungsvariante dargestellt. Durch die Programmierung mit MATLAB ergaben sich weitere vielfältige Optionen den Spielverlauf gegen den Robo-Gegenspieler interessanter zu gestalten. Die vorliegende Arbeit zeigt zunächst die bekannteste Standardumsetzung des Tic-Tac-Toe-Spiels.**

***Schlagwörter*—Farberkennung, Min-Max-Algorithmus, Roboter, Strategiespiel, Tic-Tac-Toe**

I. Einleitung

DIE Geschichte des Tic-Tac-Toe-Spiels ist in seiner Grundidee bereits auf die Zeit um 1300 v. Chr. zurückzuführen. Zwar war das Spielfeld zu dieser Zeit noch anders aufgebaut, dennoch sind die einfachen Spielregeln bis heute gleich geblieben. Da man besonders gerne Kieselsteine als Spielfiguren im Mittelalter nahm, etablierte sich der Name „Drei Kieselsteine gleichzeitig". Das Tic-Tac-Toe-Spiel, wie wir es heutzutage kennen, entstand Mitte des 19. Jahrhunderts. Durch die einfachen Spielstrukturen entstand im Jahr 1952 das erste Tic-Tac-Toe-Computerspiel mit dem Namen „OXO". Es war eines der ersten Computerspiele, bei dem der Mensch und der Computer gegeneinander spielten [1].
Nach diesem Prinzip funktioniert auch der Tic-Tac-Toe-Robo. Er funktionierte nicht über einen Bildschirm, sondern als Roboter. Er zeichnet nicht wie beim klassischem Tic-Tac-Toe das Spielfeld und seine Spielzeichen, wie es mit Zeichenroboter umgesetzt wird. Er besteht aus einem festen LEGO-Spielfeld und erkennt bzw. setzt selbstständig Spielsteine. Abbildung 1 zeigt diese vollständig verwirklichte Idee. Diese Umsetzung wird im folgenden erklärt.

II. Vorbetrachtungen

Als Spielsteine für den Roboter bieten verschiedenfarbige Kugeln mehrere Vorteile. Zu einem können sie vom LEGO-Farbsensor erkannt werden. Dadurch sollte der Farbsensor in der später folgenden Konstruktion das Spielfeld einlesen. Bei den ersten Farberkennungsversuchen wurde festgestellt, dass manche Farben nicht immer korrekt vom Sensor erkannt wurden. Dies war durch die unterschiedlichen Distanzen zwischen Sensor und Kugel, Umgebungslicht und der leicht glänzenden Oberfläche der Kugel begründet. Am sichersten erkannte der Sensor die Farben rot und grün. Aus diesem Grund wurden sie die Spielsteinfarben, wobei die grünen

DOI: 10.24352/UB.OVGU-2024-027

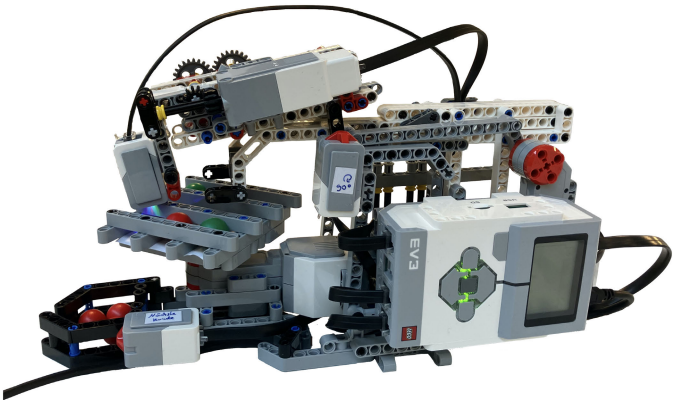

Abbildung 1. Fertiger Tic-Tac-Toe-Robo beim Einlesen des Spielfeldes

Kugeln vom Roboter und die roten vom Spieler gesetzt werden. Ein weiterer Vorteil der Kugeln zeigte sich in ihrer Form, sodass man sie ins Spielfeld fallen lassen konnte und sie die richtige Position im Feld einnehmen. Der größte Vorteil liegt jedoch im Prinzip des Auswerfens der Kugeln, da kein Greifarm benötigt wird, der die Spielsteine legen müsste.
Es war lediglich eine Lösung zu überlegen, wie man das Einlese- und Auswurfprinzip der Kugeln miteinander verknüpft und der Roboter dabei gewinnorientiert spielt. Dies realisierte der Min-Max-Algorithmus.

III. Umsetzung

Bei der Umsetzung wurde der Roboter in einzelnen Schritten aufgebaut, in denen immer neue Teilfunktionen integriert wurden. In diesem Zusammenhang wurden kleine Teilprogramme zur Ansteuerung entwickelt, die im finalen Schritt auf den gesamten Roboter angepasst wurden. Im folgenden wird der Aufbau in seinen einzelnen Schritten beschrieben. Dabei wird zwischen Konstruktion und Programmierung unterschieden.

A. Konstruktion

Die Grundvoraussetzung für den Roboter war das Spielfeld, in dem alle neun Kugeln gut saßen. Das erste Spielfeld war durch LEGO-Pins abgesteckt, wobei jede Kugel eine Fläche von ungefähr $1{,}2\,\mathrm{cm}^2$ Platz hatte. Beim Fallen der Kugeln stellte sicher heraus, dass die einzelnen Felder nicht ausreichend tief waren. Daraufhin wurde das Spielfeld erweitert und Querverstrebungen eingebaut, sodass für jede Kugel die Fläche auf $2{,}25\,\mathrm{cm}^2$ vergrößert wurde (siehe Abb. 2). Als nächstes sollte der Farbsensor über das Spielfeld zu jeder Kugel gelangen. Dies wurde mithilfe zweier Motoren realisiert. Ein Motor drehte das Spielfeld und der andere Motor ließ

einen Arm vor und zurück fahren, an dem der Farbsensor befestigt war (siehe Abb. 2). Durch die Kombination von Spielfeld drehen und vor- bzw. zurückfahren des Sensors, ist es möglich jedes Einzelfeld im Spielfeld zu erreichen.

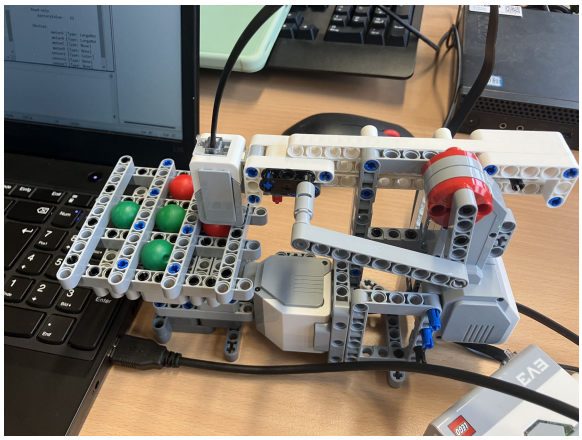

Abbildung 2. Farbsensor bestimmt Kugelfarben des gesamten Spielfelds

Nach der erfolgreichen Umsetzung des Einlesevorgangs erfolgte die Realisierung des Kugelauswurfs. Dabei saß dieser leicht geneigt auf dem Arm, an dem vorher der Farbsensor befestigt war (siehe Abb. 3). Die Neigung war notwendig, damit die nachfolgenden Kugeln aufrutschen konnten. Das Fassungsvermögen war auf fünf Kugeln ausgelegt, da dies die notwendige Anzahl an Kugeln pro Spiel ist. Der Kugelauswurf öffnete bzw. schloss sich durch ein Schließstück, das durch einen Motor betätigt wurde (siehe Abb. 3, Schließstück ist der rote Winkel).

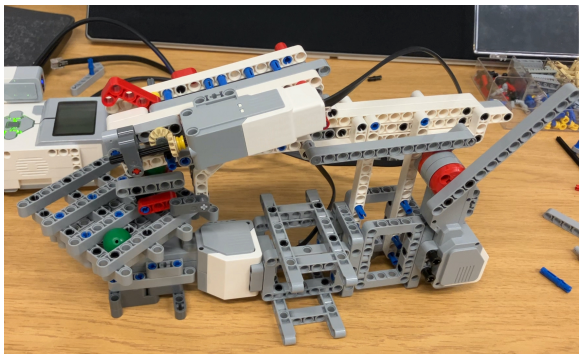

Abbildung 3. Kugelauswurf erreicht jede Spielfeldposition, Auswurfmechanismus durch Schließstück (rot) ohne Stopper

Da das Auswerfen einer Kugel notwendig war, wurde der Öffnungs- und Schließprozess schnell umgesetzt. Die hohe Schließgeschwindigkeit hatte zur Folge, dass sich eine Kugel verklemmte und sich der Winkel des Schließstücks verstellte. Dadurch fielen in den nachfolgenden Versuchen zwei Kugeln hinaus. Um dies zu vermeiden, wurde ein zusätzlicher Stopper verbaut. Dieser klinkte sich zwischen der ersten und zweiten Kugel ein, sobald sich das Schließstück öffnete. Dadurch stoppte die zweite Kugel, sodass lediglich die erste hinausfallen konnte. Der Stopper ist durch Zahnräder mit dem Schließstück verbunden. Dies hat zur Folge, dass der Stopper wieder hoch fuhr, sobald der Kugelauswurf geschlossen war. Im Anschluss rutschten die restlichen Kugeln wieder auf und der Mechanismus konnte erneut starten.
Das Gewicht von Motor und Kugelauswurf belastete den Arm stärker als der Farbsensor. Aus diesem Grund war eine Verstärkung des Arms, seiner Führung und die Unterkonstruktion des Roboters notwendig. Dabei wurde der Abstand zwischen Spielfeld und der Konstruktion, auf der die Armführung sitzt, erweitert. Dies lag an der Länge des Kugelauswurfs. Dadurch konnte er jede Position im Spielfeld erreichen.
Auch mit den Verstärkungen war ein Höhenunterschied am Arm festzustellen. Dies war in einer Toleranz zwischen Arm und Führung begründet, da der Arm etwas dünner als der Abstand zwischen der Führung war. Durch das Einbringen von zusätzlichen Stützen, die das vordere Ende der Führung erhöhten, konnte der Arm zu einer Parallele zum Spielfeld gerichtet werden (siehe Abb. 1). Am meisten profitierte der Farbsensor von dieser neuen Ausrichtung, der im Anschluss vor dem Kugelauswurf angebaut wurde. Somit ist die Distanz zwischen Kugel und Sensor immer gleich, sodass benannte Fehler bei der Farberkennung (erste Farberkennungsversuche in Abschnitt II) vermieden wurden.
Da es teilweise beim Treffen der Felder zu Fehlpositionen kam, wurde um die Fallstrecke eine Bande errichtet, die ein Wegrollen der Kugeln verhinderte. Im Weiteren ergänzten zwei Taster den Roboter. Der untere Taster diente als Start-Befehl für den Roboter zum Einlesen, Auswurfposition berechnen und Auswerfen der Kugeln. Der obere Taster ermöglichte das Spielfeld um 90° zu drehen (siehe Abb. 1). Dies war notwendig, da das vordere Einzelfeld vom Farbsensor verdeckt war. Dies erleichterte das Legen sowie das Herausnehmen der Kugeln. Im finalen Schritt wurde der Roboter durch ein Behältnis für die roten Kugel ergänzt und optisch angepasst, wie beispielsweise die Kabelführung.

B. Programmierung

Im folgenden wird der Aufbau des Programms erklärt. Die Erklärung wird vereinfacht durch den Programmablaufplan in Abbildung 4 dargestellt.
Alle eigenständigen Abläufe des Roboters wurden in einzelne Funktionen untergliedert, wie beispielsweise das Auswerfen einer Kugel. Diese sind im Hauptprogramm miteinander verknüpft. Als ersten Schritt wird im Hauptprogramm die Steuereinheit EV3 neu initialisiert. Dadurch konnten Fehlfunktionen mit der Winkelbegrenzung und Motorbremse vermieden werden. Im nächsten Schritt wurden für die Funktionen alle Winkel und Geschwindigkeiten der Motoren festgelegt. Das erfolgt über Variablen, da sie universell einsetzbar sind und Fehler reduzieren, bei den Anpassungen dieser Werte. Andernfalls wäre diese Anpassung in jeder einzelnen Funktion notwendig. Nachdem alle Variablen durch Werte bestimmt und der Farbsensor initialisiert ist, konnte eine 3×3-Matrix erstellt wurden. Dies entspricht der Einsmatrix, da die 1 im Späteren für ein nicht belegtes Feld steht. Die Matrix stellt das Spielfeld dar.

Durch das betätigen des unteren Tasters, wurde der Roboter gestartet. Folgend wurde jeder Motor gestoppt um Fehler zu vermeiden. Mögliche Fehler zeigten sich bei ersten Versuchen, indem ein Motor nicht seine Drehwinkelbegrenzung erreichte. Dabei versuchte der Motor sie weiterhin zu erreichen, auch wenn er mechanisch gehindert wird und stoppte somit nie. Da jedoch auf den Stillstand des Motors gewartet wird, führte dies zum Stillstand des gesamten Programms. Im Anschluss wird für die folgenden Spielrunden in allen Motoren die Motorbremse und der Drehwinkel begrenzt. Das war notwendig, da sich sonst die Ausgangsposition in jeder Spielrunde änderte.

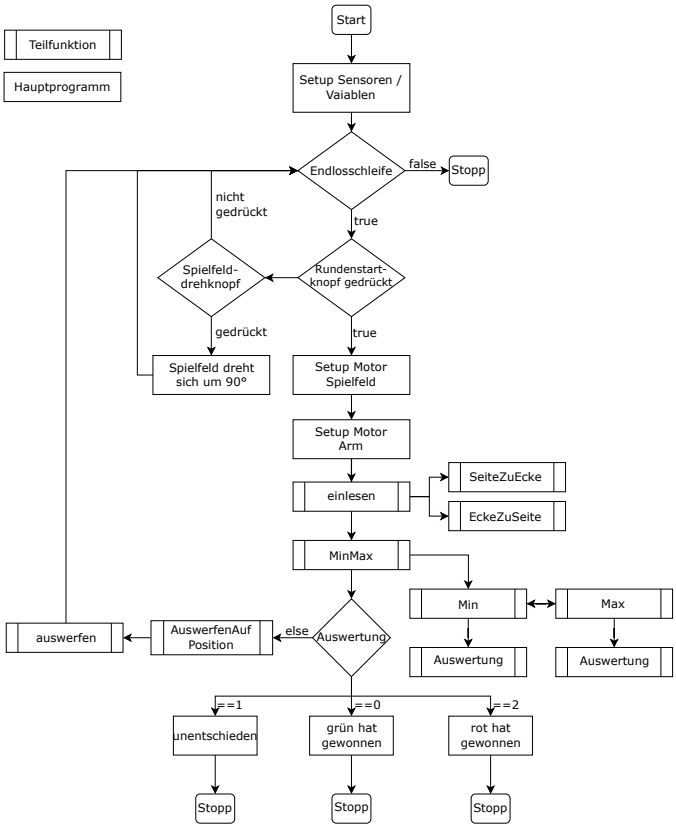

Abbildung 4. Programmablaufplan

Im Anschluss startet der Roboter mit dem Einleseprozess, der durch eine Funktion beschrieben ist. Sobald der Farbsensor eine Kugelfarbe im Spielfeld entdeckte, überschrieb er die erwähnte 3×3-Matrix mit der ermittelten Konstellation. Dabei nahm die Stelle in der Matrix den Farbsensorwert an, die der Kugelfarbe entspricht. Somit ist es möglich jeden Feldzustand durch eine Zahl zu beschrieben. Hierbei entsprach die Kugelfarbe grün dem Wert 3 und die rote Kugel dem Wert 5. Ein leeres Feld entsprach dem Farbsensorwert 1 (grau). Die Einlesefunktion wurde durch zwei weitere Funktionen ergänzt, die das Abfahren von einer Kanten- zu einer Eckposition (SeiteZuEcke) und von einer Eck- zu einer Kantenposition (EckeZuSeite) beinhalten.
Nach dem erstellen einer Matrix, die dem Spielstand entsprach, musste ein Algorithmus entstehen, der die Auswurfposition berechnet. Dabei war eine kurzzeitige Überlegung, alle möglichen Spielausgänge einzuprogrammieren. Nach einer kurzen Aufwandsabschätzung wurde diese Idee verworfen, da es $9! = 362880$ mögliche Spielverläufe gibt. Auch wenn die möglichen Spielverläufe durch Nutzung der Symmetrie und ein frühes Gewinnen reduziert werden, wäre die Zeit zu knapp gewesen, um alle möglichen Pfade zu programmieren. Eine weitere Möglichkeit bot das Addieren der einzelnen Zeilen, Spalten und Diagonalen, um im Anschluss einen Algorithmus dafür zu programmieren. Allerdings gab es bei allen probierten Zahlenkombinationen, bei mindestens zwei unterschiedlichen Szenarien die gleichen Ergebnisse. Ein Beispiel hierfür ist, mit den bekannten Zahlen 1, 3 und 5, die zwei Kombinationen $3 + 3 + 3 = 9$ (grün + grün + grün) und $1 + 3 + 5 = 9$ (leeres Feld + grün + rot). Nach einer Recherche bietet sich der Ansatz des Min-Max-Algorithmus an. Zur detaillierten Einarbeitung waren zwei Videos von Tobias Nopper [2],[3] hilfreich.

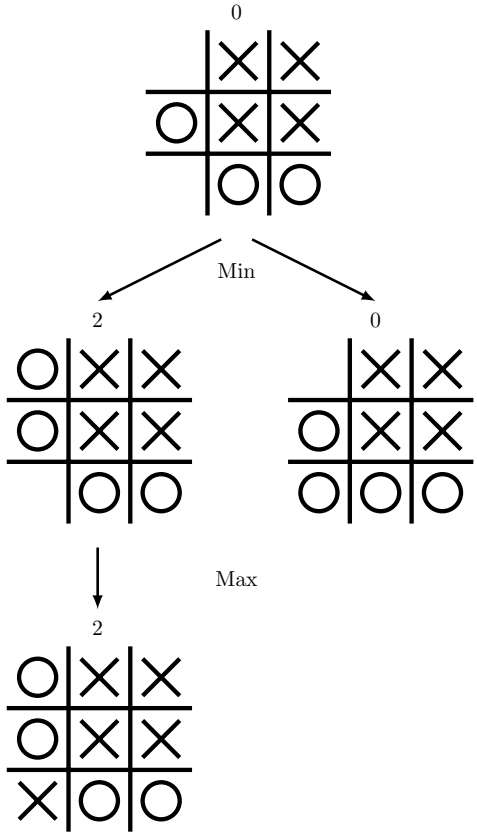

Abbildung 5. Min-Max-Algorithmus anhand eines Beispiels

Der Min-Max-Algorithmus wird mithilfe zweier rekursiver Funktionen umgesetzt. Wenn der Roboter gewinnt, wird dies durch ein Minimalwert beschrieben und wenn der Spieler gewinnt durch einen Maximalwert. Gewinnt keiner, wird ein Wert zwischen Minimum und Maximum festgelegt. Die Funktionen werden immer abwechselnd abgerufen und geben der anderen Funktion ihren Maximal- bzw. Minimalwert weiter. Als Beispiel dient Abbildung 5. Hierbei gibt es zwei Möglichkeiten. Zu einem kann „O" gewinnen, wobei sich der Minimalwert 0 ergibt. Zum anderen kann „X"

gewinnen, wobei sich der Maximalwert 2 ergibt. Hier wird eine Minimalauswertung für das obere Feld bestimmt, da „O" am Spielzug ist. Weil 0 kleiner als 2 ist, wird für das obere Feld die 0 weitergegeben. Somit sagt die 0 aus, dass „O" im nächsten Spielzug gewinnt. Über diese Methode werden alle möglichen Pfade berechnet und ihre Gewinnchancen bestimmt. Damit der Roboter seine Auswurfposition bestimmen kann, wird der Minimalwert abgefragt. Dabei gibt die Funktion alle Positionen aus, die einen Minimalwert anstreben. Hierbei wird der Roboter die letzte Position verwenden, da sie die vorherigen Positionen überschreibt. Aus diesem Grund unterscheidet der Algorithmus nicht wie schnell man den Sieg erreicht. Das bedeutet, dass der Roboter mit einem Spielzug gewinnen könnte, jedoch woanders hinsetzt und später gewinnt.
Nachdem eine günstige Position gefunden wurde, richtete sich der Kugelauswurf über diese Feldpostion aus. Diese Ausrichtung ist in einer eigenen Funktion beschrieben. Sie ordnet jeder berechneten Postion die Ausrichtung des Kugelauswufs zu. Wenn der Kugelauswurf die richtige Position erreicht hat, startet die Auswerfenfunktion. Im finalen Schritt wurde mithilfe der Stellmotoren die Ausgangsposition angestrebt.
Das Teilprogramm, dass durch den Tasters ausgelöst wurde ist damit abgeschlossen. Nun konnte der Spieler setzten oder den anderen (oberen) Taster für die Spielfeld-Drehung um 90° drücken. Durch das wiederholte drücken des unteren Taster startet das Teilprogramm erneut. Stellt der Roboter beim Einlesevorgang fest, dass er gewinnt, gibt er eine akustische Meldung ab und dreht dabei das Spielfeld drei mal. Wenn er verliert bzw. keiner gewinnt, gibt er andere akustische Meldungen aus.

IV. Ergebnisdiskussion / Verbesserungsvorschläge

Die Motoren, die für die Drehung des Spielfeldes und der Vor- bzw. Zurückbewegung des Armes zuständig sind, haben eine mechanische Toleranz. Dies hat zur Folge, dass das Spielfeld und der Arm für ein reibungslosen Ablauf perfekt zueinander ausgerichtet sein müsste. Die Fehleranfälligkeit des Farbsensors ist höher als die vom Kugelauswurf, da er eine Bande hat. Aus diesem Grund wäre eine Selbstkalibrierung beispielsweise durch farbigen Linien vorteilhaft. Alternativ könnte auch eine Bilderkennung mittels Kamera verwendet werden. Dabei wird kontinuierlich das ganze Feld erfasst. Durch diese Erkennung könnte der Roboter auch ein Schummeln des Spielers oder einen Fehler beim Auswurf der Kugeln feststellen.
Wenn der Roboter setzt und damit gewinnt, wäre es eine Verbesserung, wenn er direkt in den Gewinnmodus übergeht, ohne nochmal das Spielfeld einzulesen. Eine weitere Optimierung erfolgt, wenn der Roboter mit so wenig wie nötigen Spielzügen gewinnt. Im Spiel selbst traten keine weiteren Probleme auf. Eine letzte Optimierung ist durch die Nutzung von MATLAB möglich, da es seine eigene `Minimax` Funktion hat. Durch ihr bräuchte man lediglich diese eine Funktion anstelle von zwei rekursive Einzelfunktionen.

V. Zusammenfassung / Fazit

Die Erwartungen, einen Roboter zu bauen, der gegen einen menschlichen Spieler gewinnorientiert Tic-Tac-Toe spielt, wurde erfolgreich umgesetzt. Es wurden für den Roboter drei Motoren, ein Farbsensor, zwei Taster und als Steuereinheit der EV3 verwendet. Die Hauptbestandteile des Programms sind das Spielfeld einzulesen, auszuwerten und eine Position zum Setzten zu berechnen und schlussendlich genau in diese Position die Kugel auszuwerfen.
Für zukünftige Tic-Tac-Toe-Spiele, die über einen ähnlichen Aufbau verfügen, sind folgende Optimierungen empfehlenswert:

1) eine Selbstkalibrierung oder Bildauswertung mithilfe einer Kamera
2) so wenig Züge wie nötig zum Gewinnen
3) Erkennung falls der Spieler schummelt oder die Kugel falsch fällt
4) die eigene MATLAB-Funktion `Minimax` verwenden
5) Gewinnen erkennen ohne ein erneutes Einlesen

Auch für zukünftige Strategiespiele ist der Min-Max-Algorithmus zu empfehlen, da er sehr komplizierte Verzweigungen von Pfaden erleichtert sowie die Gewinnchancen und mögliche Setztposition bestimmt.

Literaturverzeichnis

[1] Marcus, M.: Tic Tac Toe-Geschichte: Drei In Folge Im Wandel Der Zeit, URL: *https://www.coolmathgames.com/de/blog/tic-tac-toe-geschichte-drei-in-folge-im-wandel-der-zeit* (Stand: 2024-02-16)

[2] Nopper, T.: Computer spielen: Wie eigentlich?: Der Min-Max-Algorithmus, URL: *https://www.youtube.com/watch?v=3ufcmCpKb6w* (Stand: 2024-02-07)

[3] Nopper, T.: Computer spielen: Wie eigentlich?: MinMax implementieren für Tic Tac Toe, URL: *https://www.youtube.com/watch?v=ODgPsXssUvk* (Stand: 2024-02-07)