

Billiard-AI

Ein intelligenter Billardtisch

BSc Thesis

Studienrichtung:

Autor:

Dozent:

Experte:

Datum:

Informatik - Computer Perception and Virtual Reality

Lukas Seglias, Luca Ritz

Markus Hudritsch

27. September 2021

Inhaltsverzeichnis

| | |
|---|-----------|
| 1 Zusammenfassung | 1 |
| 2 Einführung | 3 |
| 3 Ziele | 5 |
| 3.1 Planung | 6 |
| 3.1.1 Meilensteine | 6 |
| 4 Billiard-AI | 9 |
| 4.1 Vorarbeiten | 9 |
| 4.2 Risikoanalyse | 10 |
| 4.3 Klassifikation | 11 |
| 4.4 Suchalgorithmus | 12 |
| 4.4.1 Kandidatensuche | 12 |
| 4.4.2 Berechnung der Initialgeschwindigkeit | 13 |
| 4.4.2.1 Reibungsverlust über Bahn | 14 |
| 4.4.2.2 Energieübergabe bei Kugelkollision | 14 |
| 4.4.3 Modellierung physikalisches System | 15 |
| 4.4.3.1 Ereignisse und ihre Repräsentation | 16 |
| 4.4.3.2 Kantenfunktion | 17 |
| 4.4.3.3 Layer | 17 |
| 4.4.3.4 Beispiel eines Graphen | 17 |
| 4.4.4 Simulation | 18 |
| 4.4.4.1 Reibungsverlust über Strecke | 18 |
| 4.4.4.2 Ereignis Out-Of-Energy | 18 |
| 4.4.4.3 Ereignis Energy-Transfer über Kugelkollision | 19 |
| 4.4.4.4 Ereignis Energy-Transfer über Bandenkollision | 19 |
| 4.4.4.5 Ereignis Out-Of-System | 19 |
| 4.4.5 Animation | 19 |
| 4.5 Berechnungsprozess | 20 |
| 5 Resultate | 23 |
| 5.1 Klassifikation | 23 |
| 6 Weitere Arbeiten | 25 |
| 7 Fazit | 27 |
| 8 Anhang | 37 |

1 Zusammenfassung

TODO: Zusammenfassung

2 Einführung

Wie vieles andere ist auch das Erlernen des Billardspiels eine schwierige Sache. Es stellen sich Fragen wie „Welche Kugel soll man anspielen?“, „Wie soll man die Kugel anspielen?“ oder „Wie hält man den Queue richtig?“. Darauf folgend gibt es noch diverse weitere Überlegungen, welche den Profi vom Anfänger unterscheiden. Wie in anderen Spielen auch, ist hier Weitsicht gefragt. Es geht also nicht nur darum, eine Kugel zu versenken, sondern auch den Spielstand so zu verändern, dass optimal weitergespielt werden kann. Das Stichwort ist im Billard vor allem die Platzierung der weißen Kugel.

TODO: Weiter schreiben

3 Ziele

Ins Billard-Spiel einzusteigen ist nicht ganz einfach. Zu Beginn lässt es sich schlecht abschätzen, welchen Weg eine Kugel nehmen wird, wenn man sie anschlägt und den optimalen Stoss über mehrere Züge hinaus zu planen, erst recht. Denn bei fortgeschrittenen Spielen ist es oft wichtig, dass die weisse Kugel optimal für den nächsten Stoss platziert wird.

Es soll ein System entstehen, das dem Spieler den optimalen Stoss vorschlägt basierend auf Kriterien und optional einer festgelegten Tiefe des Spielstands. Dazu soll eine Kamera den Spielstand auf dem Billiartisch erkennen, verarbeiten und dem Spieler Hilfestellungen mittels eines Projektors anzeigen.

Diese Arbeit setzt auf bereits geleisteter Tätigkeit aus „Projekt 2“ auf[Luk21a]. Die Tätigkeiten werden nachfolgend beschrieben.

Vorschlag eines optimalen Stosses Es wird ein Stoss vorgeschlagen, welcher anhand der gewählten Kugel möglichst optimal ist. Dieser Stoss kann direkt (einfach) oder indirekt (erweitert) sein.

Es ist weiterhin anzumerken, dass es in erster Linie um Snooker-Billard geht. Dies hat mehrere Gründe. Einerseits soll in dieser Arbeit nicht die Klassifikation der Kugeln im Zentrum stehen, sondern die Suche nach einem optimalen Stoss. Es wird angenommen, dass dies mit Snooker-Kugeln einfacher ist als mit Pool-Billard-Kugeln. Andererseits wird das Projekt zusammen mit einem Unternehmen durchgeführt, welches eventuell auch einen kommerziellen Ansatz verfolgen will. Da grössere Turniere wie Weltmeisterschaften in Snooker ausgetragen werden, kam schnell der Wunsch auf, das Hauptaugenmerk darauf zu legen. Nichtsdestotrotz wird die Anwendung so abstrakt gehalten, dass sie mit wenig Aufwand auf Pool-Billard portiert werden könnte. Dies bildet jedoch kein Ziel der Bachelor-Thesis.

3.1 Planung

Die initiale Planung beinhaltet eine Auflistung der Tätigkeiten, dem zugewiesenen Meilenstein sowie deren Schätzung in PT (Personen-Tage). Jedem Arbeitspaket wird eine ID zugewiesen, welche bei der Zeiterfassung verlinkt wird. Das Total der zu vergebenden PT beträgt 90.

| ID | Name | Meilenstein | Schätzung in PT |
|-------|--|-------------|-----------------|
| T-1 | Klassifikation der Kugeln | M-1 | 6 |
| T-2 | Aufsetzen Dokumentation | M-1 | 2 |
| T-3 | Beschreibung Suchalgorithmus | M-1 | 3 |
| T-4 | Implementation Suchalgorithmus | M-1 | 5 |
| T-5 | Beschreibung der physikalischen Eigenschaften für die einfache Suche | M-1 | 6 |
| T-6 | Implementation der einfachen Suche und deren Bewertungsfunktion | M-1 | 10 |
| T-7 | Beschreibung der physikalischen Eigenschaften für die erweiterte Suche | M-2 | 6 |
| T-8 | Implementation der erweiterten Suche und deren Bewertungsfunktion | M-2 | 8 |
| T-9 | Überprüfen/Verbessern der Detektionsgenauigkeit | M-1 | 6 |
| T-10 | Video erstellen | M-3 | 2 |
| T-11 | Plakat schreiben | M-3 | 2 |
| T-12 | Booklet-Eintrag schreiben | M-3 | 1 |
| T-13 | Präsentation des Finaltags vorbereiten | M-3 | 2 |
| T-14 | Präsentation der Verteidigung vorbereiten | M-3 | 2 |
| T-15 | Finalisieren Dokumentation andere Arbeiten | M-3 | 4 |
| T-16 | Projektmanagement | Kein | 4 |
| T-17 | Effizienz Erfassung und Steigerung der einfachen Suche | M-1 | 4 |
| T-18 | Effizienz Erfassung und Steigerung der erweiterten Suche | M-2 | 2 |
| T-19 | Dokumentation der Resultate der einfachen Suche | M-1 | 6 |
| T-20 | Dokumentation der Resultate der erweiterten Suche | M-2 | 2 |
| T-21 | Umbau in Unity | M-1 | 6 |
| O-1 | Suche über mehrere Spielstände | M-2 | |
| O-2 | Detektion des Queues in 2D | M-2 | |
| O-3 | Detektion des Queues in 3D | M-2 | |
| O-4 | Stossberechnung anhand detektiertem Queue in 2D | M-2 | |
| O-5 | Stossberechnung anhand detektiertem Queue in 3D | M-2 | |
| O-6 | Spielerabhängige Heuristik | M-2 | |
| O-7 | Live-Verfolgung und Darstellung der Kugeln | M-2 | |
| Total | | | 90 |

Tabelle 3.1: Ziele

3.1.1 Meilensteine

Es werden drei Meilensteine definiert, welche auch aus optionalen Zielen bestehen können. Die Deadlines ergeben sich aus den Schätzungen der zugewiesenen Arbeitspakete.

Meilenstein 1 - 15.11.2021 Das Ziel ist eine sehr einfache simple Suche. Darunter zu verstehen ist eine Lösung, welche einen direkten Treffer findet (Weiss -> Kugel -> Loch).

Code Deliverables:

Klassifikation - T-1

Alle Kugeln können entsprechend ihre Farbe klassifiziert werden.

Suchalgorithmus für einfache Suche - T-4, T-6, T-17

Ein direkter Stoss wird in akzeptabler Zeit gefunden.

Unity-Umbau - T-21

Unity ist bereit für den Einsatz. Zum Umbau gehören insbesondere die Farbe der Markierung der Kugeln und deren Bahnen. Weiterhin muss Unity mehrere Suchergebnisse anzeigen können.

Dokumentation Deliverables:

Klassifikation - T-1

Das Vorgehen der Klassifikation wie deren Resultate und Genauigkeit sind dokumentiert.

Suchalgorithmus für Suche - T-3

Der Algorithmus der Suche ist theoretisch und mit Pseudocode beschrieben. Die theoretische Beschreibung muss nicht gänzlich mit der effektiven Implementation übereinstimmen, da diese auf Performance optimiert wird.

Resultate der einfachen Suche - T-19

In den Resultaten ist die Genauigkeit und Performance des einfachen Suchvorgangs beschrieben.

Physik der einfachen Suche - T-5

Die benötigte Physik der einfachen Suche ist beschrieben.

Bewertungsfunktion - T-6

Die Bewertungsfunktion der einfachen Suche ist dokumentiert.

Meilenstein 2 - 13.12.2021 Das Ziel ist eine erweiterte Suche, die auch indirekte Stöße über weitere Kugeln oder Banden finden kann. Optional sollen auch mehrere Stöße berücksichtigt werden.

Code Deliverables:

Suchalgorithmus für erweiterte Suche - T-8, T-18

Ein indirekter Stoß wird in akzeptabler Zeit gefunden.

Suchalgorithmus über mehrere Stöße - O-1

Es werden mehrere Spielstände bei der Suche berücksichtigt.

Queue in 2D detektieren - O-2

Der Queue wird als 2D-Objekt detektiert.

Queue in 3D detektieren - O-3

Der Queue wird mittels Tiefeninformationen der Kamera als 3D-Objekt detektiert.

Stoßberechnung anhand detektiertem 2D-Queue - O-4

Der Stoß wird je nach Haltung des Queues in 2D berechnet. Es wird angenommen, dass der Queue zentral auf die weiße Kugel gerichtet ist.

Stoßberechnung anhand detektiertem 3D-Queue - O-5

Der Stoß wird je nach Haltung des Queues in 3D berechnet. Der Queue muss nicht mehr zentral auf die weiße Kugel gerichtet sein.

Spielerabhängige Heuristik - O-6

Je nach Spieler kann eine andere Heuristik zur Bewertung der Stöße eingestellt werden. Durch die Unterscheidung können für professionelle Spieler erfolgsversprechendere schwerer durchzuführende und für Anfänger eher leichtere Stöße gefunden werden.

Live-Verfolgung und Darstellung der Kugeln - O-7

Die Kugeln werden ohne Benutzereingabe getrackt und deren Position über den Projektor dargestellt.

Dokumentation Deliverables:

Physik der erweiterten Suche - T-7

Die benötigte Physik der erweiterten Suche ist beschrieben.

Resultate der erweiterten Suche - T-20

In den Resultaten ist die Genauigkeit und Performance des erweiterten Suchvorgangs beschrieben.

Bewertungsfunktion - T-8

Die Bewertungsfunktion der erweiterten Suche ist dokumentiert.

Meilenstein 3 - 17.01.2022 Das Ziel ist der Abschluss aller Arbeiten zu denen auch Plakat, Booklet oder Video gehören.
Deliverables:

Video - T-10

Das finale Video ist erstellt.

Plakat - T-11

Das Plakat ist erstellt.

Booklet - T-12

Der Booklet-Eintrag ist erstellt.

Präsentation für Finaltag - T-13

Die Präsentation/Ausstellung für den Finaltag ist vorbereitet.

Präsentation für Verteidigung - T-14

Die Präsentation für die Verteidigung ist vorbereitet.

Finalisieren der Arbeiten - T-15

Die Dokumentation wie auch der Code sind abgeschlossen.

4 Billiard-AI

4.1 Vorarbeiten

Zu dieser Bachelor-Thesis gab es bereits eine Vorarbeit im letzten Semester. Diese hatte zum Ziel, einige grundlegende Funktionalitäten bereitzustellen. In Abbildung 4.1 ist der Zyklus angegeben, welcher alle Aufgaben zusammenfasst. In einem ersten Schritt wird der aktuelle Spielstand über eine Kamera detektiert. Daraus kann eine textuelle Beschreibung abgeleitet werden. Diese dient wiederum als Input für das Kernstück, die Suche nach einem optimalen Stoss. Als Ausgabe dieses Schrittes erfolgt eine detaillierte Animation des auszuführenden Stosses. Die Animation wird über einen Projektor auf dem Billardtisch dem Spieler zugänglich gemacht. Dieser kann den Stoss ausführen, was zu einer neuen Situation führt und der Zyklus beginnt von vorne.

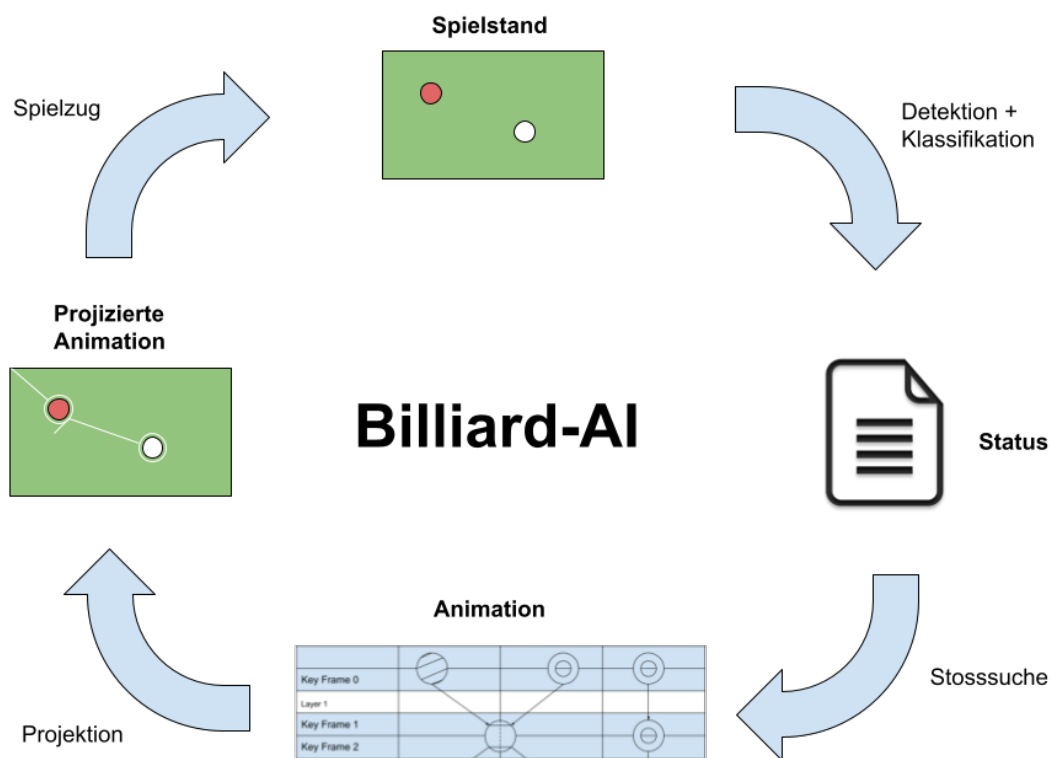


Abbildung 4.1: Billard-AI-Cycle

Es sind nun die Schritte der Detektion des Spielstands wie auch die Projektion und Darstellung der Animation auf dem Spieltisch bereits umgesetzt. Dabei ging es in erster Linie um die Frage der Repräsentation in geeignetem Koordinatensystem sowie die Messung der Genauigkeit dieser Übersetzung und der darauf basierenden Anzeige über den Projektor.

Die Bachelor-Thesis widmet sich zuerst der Frage nach der Klassifikation in Schritt eins, wie auch der Suche nach dem optimalen Stoss in Schritt zwei.

4.2 Risikoanalyse

Es gibt diverse Risiken, die während dieser Arbeit eintreten können. Um das Bewusstsein dafür zu stärken, wird vorgängig eine Risikoanalyse durchgeführt, wobei es um die Identifikation wie auch die Zuordnung deren Auftretenswahrscheinlichkeit und Auswirkungen geht. Weiterhin werden geeignete Massnahmen definiert, die entweder die Eintrittswahrscheinlichkeit reduziert oder bei Auftreten angegangen werden können.

Die Risiken werden im Detail aufgelistet. Die Spalte „WK“ steht für die Eintrittswahrscheinlichkeit, welche in den Wahrscheinlichkeiten „Gering“, „Möglich“, „Wahrscheinlich“, „Sehr Wahrscheinlich“ angegeben wird. Die Spalte „AW“ steht für die Auswirkungen, welche in den Grössen „Klein“, „Mittel“, „Gross“ angegeben wird.

| ID | Risiko | Massnahme | WK | AW |
|-----|--|---|----------------|--------|
| R-1 | Verlust von Programm-Sourcen | Führen eines Repositories auf GIT, welches das Wiederherstellen eines bestimmten Standes erlaubt. Zudem wird jeden Freitag ein Backup des GIT-Standes auf eine externe Festplatte geschrieben, sollte der unwahrscheinliche Fall eintreten, dass GIT nicht mehr verfügbar sein sollte oder seine Bestände verliert. | Möglich | Mittel |
| R-2 | Ausfall der Arbeitsgeräte | Es stehen Ersatzgeräte bereit, welche sofort zum Einsatz kommen könnten. | Möglich | Klein |
| R-3 | Krankheitsausfall der Teammitglieder | Es wird wenn möglich von Zuhause aus gearbeitet, um das Risiko einer Ansteckung zu vermindern. | Möglich | Gross |
| R-4 | Parallele Entwicklung derselben Funktionen | Durch eine anfängliche Planung der Arbeitspakete, wie auch den ständigen Austausch und Einsatz von Pair-Programming an geeigneten Stellen, wird das Risiko stark reduziert. Sollte es trotzdem Eintreten, sind die Auswirkungen marginal, da die ständige Kommunikation dies sofort aufdecken würde. | Gering | Klein |
| R-5 | Unterschätzen der Komplexität | Um zumindest ein brauchbares Resultat vorweisen zu können, wurde der erste Meilenstein möglichst simpel gehalten. | Wahrscheinlich | Gross |
| R-6 | Verpassen wichtiger Termine | Es wird ein Kalender mit allen Terminen geführt, welcher mehrmals eine Erinnerung anzeigt. | Gering | Gross |

Tabelle 4.1: Risiken

Die identifizierten Risiken werden in der Abbildung 4.2 für eine bessere Übersicht eingetragen.



Abbildung 4.2: Risikoanalyse

4.3 Klassifikation

4.4 Suchalgorithmus

Die Suche wird über zwei Schritte durchgeführt. Der erste Schritt besteht aus der Suche nach einem Lösungskandidaten, wobei vom Ziel aus ein Stoss gesucht wird, welcher das Potenzial hat, eine Kugel in diesem Ziel zu versenken. Das Resultat dieses Schrittes ist lediglich der Geschwindigkeitsvektor der weissen Kugel. Ob dieser Stoss tatsächlich das Resultat zur Folge hat, welches er voraussagt, wird im zweiten Schritt geprüft. Der Geschwindigkeitsvektor der weissen Kugel kann mit unterschiedlichem Betrag in einen Simulationsschritt eingegeben werden. Der Simulationsschritt wird als Resultat ein physikalisches System wie in Kapitel 4.4.3 ergeben.

Im Nachfolgenden wird auf die verschiedenen Schritte und deren Funktionsweise sowie die optimale parallele Durchführung der Berechnungen eingegangen.

4.4.1 Kandidatensuche

Die Kandidatensuche, im Folgenden Suche genannt, wird über eine klassische Graphensuche durchgeführt, wobei der vollständige Graph alle möglichen Stösse enthält. Für die Suche gibt es zwei Möglichkeiten, entweder wird bei der weissen Kugel gestartet und von dort ein Stoss gesucht, welcher eine andere Kugel ins Loch spielt, oder es wird bei einem oder mehreren Löchern gestartet und von dort ein Stoss gesucht, welcher von der weissen Kugel ausgehend eine andere Kugel ins Loch spielt.

Nachfolgend wird die Suche beschrieben, welche beim Loch, dem Ziel, startet, eine einzulochende Kugel findet und anschließend den Stoss bis zur weissen Kugel zurück sucht. Dementsprechend ist der Root-Knoten des Suchbaumes das zu treffende Ziel (Loch). Da ein handelsüblicher Billardtisch mehrere Löcher hat, muss pro Loch eine separate Suche durchgeführt werden.

Bei der Durchführung eines Expansionsschrittes werden ausgehend von einem Knoten im Suchbaum dessen Nachfolger-Knoten ermittelt. Diese stellen im Fall vom Root-Knoten Kugeln dar, welche in dieses Loch gespielt werden könnten. Aus diesen Kugeln werden Kugel-Knoten gebildet, welche diese Kugeln entweder auf direktem Wege oder indirekt über die Bande in das Loch spielen lassen sollen. Ausgehend von diesen Kugel-Knoten, werden deren Nachfolger-Knoten in weiteren Expansionsschritten ermittelt, welche wiederum Kugel-Knoten darstellen. Diese Kugel-Knoten stellen dann Kugeln dar, welche die Kugel des Vorgänger-Kugel-Knotens entweder auf direktem Wege oder indirekt über die Bande treffen sollen. Sofern ein Kugel-Knoten die weisse Kugel darstellt, so ist dieser Kugel-Knoten ein Endzustand und damit ist der Stoss über die Kette von Nachfolger- zu Vorgänger-Kugel-Knoten definiert.

Zur Veranschaulichung des Prinzips folgt ein Beispiel. Es wird vereinfacht angenommen, dass der Tisch nur ein Loch hat. Für mehrere Ziele ergeben sich mehrere Suchbäume, einen pro Loch. In Abbildung 4.3 erfolgt die Eingabe des Suchalgorithmus in Form des Root-Knotens. Es wird nur das zu treffende Ziel definiert. Auf der rechten Seite des Tisches ist der Suchbaum dargestellt.

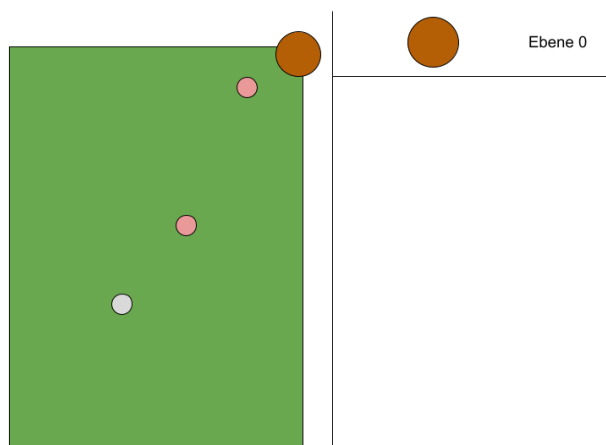


Abbildung 4.3: Kandidatensuche 1

In einem zweiten Schritt wird die einzulochende Kugel definiert. Es kommen lediglich die beiden roten Kugeln in Frage. Nachfolgend wird der Pfad weiter betrachtet, bei dem die rote Kugel, welche näher beim Loch ist, gewählt wurde. Abbildung 4.4 zeigt, dass der Suchbaum um einen Knoten erweitert wurde.

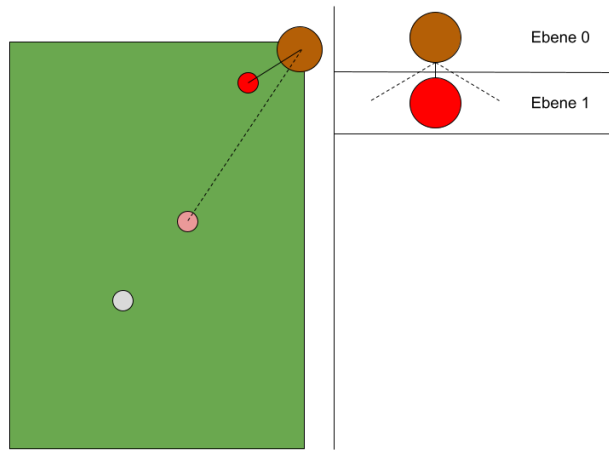


Abbildung 4.4: Kandidatensuche 2

In Abbildung 4.5 erfolgt der letzte Schritt. Hier sind verschiedene Optionen möglich, bspw. könnte die weiße Kugel direkt oder über die Bande an die zuvor gewählte rote Kugel gespielt werden. Es könnte aber auch die andere rote Kugel an die zuvor gewählte rote Kugel gespielt werden. Hier wird der Fall betrachtet, dass die weiße Kugel indirekt über eine Bande an die zuvor gewählte rote Kugel gespielt wird.

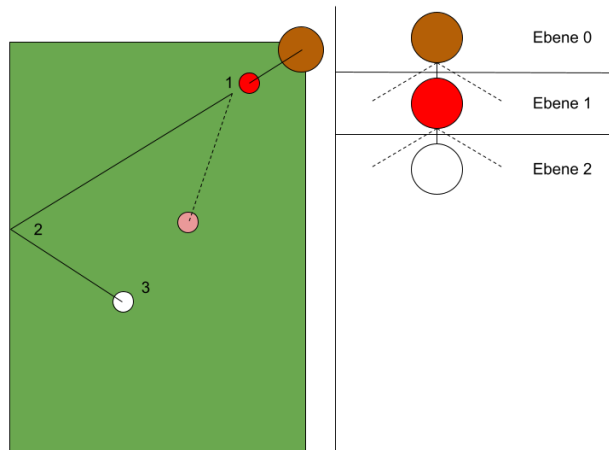


Abbildung 4.5: Kandidatensuche 3

Algorithmus 1 verdeutlicht den ablauf der „Expand-Funktion“. Zuerst wird eine leere Liste namens „nodes“ angelegt. Diese wird danach mit Nodes gefüllt, welche entweder durch einen Stoss über eine weitere Kugel oder indirekt über die Bande zustande kommen. Die Nodes bilden das Ergebnis der Funktion.

```

Function expand(node: Node, constantObjects: list) → list[Node]
    nodes ← list()
    nodes ← append(expandBalls(node, constantObjects), nodes)
    nodes ← append(expandBank(node, constantObjects), nodes)
    return nodes

```

Algorithm 1: Algorithmus zur Durchführung eines Expansionsschritts bei der Kandidatensuche

4.4.2 Berechnung der Initialgeschwindigkeit

Sofern ein Stosskandidat wie in Abschnitt 4.4.1 beschrieben gefunden wurde, muss berechnet werden, welche Geschwindigkeit die weiße Kugel erhalten muss, damit die gewünschte Kugel ins Loch gespielt wird. Relevant dafür ist der Reibungsverlust wenn die Kugel rollt und die Kollision mit anderen Kugeln. Die Berechnungen sind nachfolgend beschrieben.

4.4.2.1 Reibungsverlust über Bahn

TODO: T-5

4.4.2.2 Energieübergabe bei Kugelkollision

TODO: T-5

4.4.3 Modellierung physikalisches System

Ein Stoss kann für die Simulation mithilfe eines Modells beschrieben werden. Dieses Modell beschreibt den Ablauf eines Stosses durch Ereignisse, welche den beteiligten Objekten widerfahren. Die Kugeln, Banden und Löcher sind die Objekte dieses Modells. Jedes Objekt dieses Modells hat einen veränderlichen Energiewert, bspw. sind Kugeln in Bewegung oder sie sind ruhend. Eine Bande oder ein Loch bewegen sich nie, d.h. deren Energiewert ist konstant 0. Kollisionen zweier Kugeln oder einer Kugel und der Bande sind Beispiele von Ereignissen, welche Veränderungen der Energiezustände der Objekte bewirken.

Objekte, die einen Energiewert besitzen, welcher sich über die Zeit oder durch Interaktionen mit anderen Objekten verändert, werden nachfolgend variabel genannt. Sofern ein variables Objekt einen Energiewert grösser 0 hat, dann wird es weiter als dynamisch bezeichnet. Sobald es den Energiewert 0 erreicht, ist es als statisches variables Objekt kategorisiert. Banden und Löcher, welche einen fixen Energiewert besitzen, werden als konstant beschrieben.

Zwischen den Ereignissen kann eine Energieabnahme stattfinden. Beispielsweise unterliegt eine rollende Kugel der Rollreibung und verliert dadurch bis zum nächsten Ereignis Energie. Diese Energieabnahme erfolgt durch die Anwendung der Kantenfunktion auf der Kante zwischen zwei Ereignissen.

Diese Nomenklatur ist in Abbildung 4.6 ersichtlich.

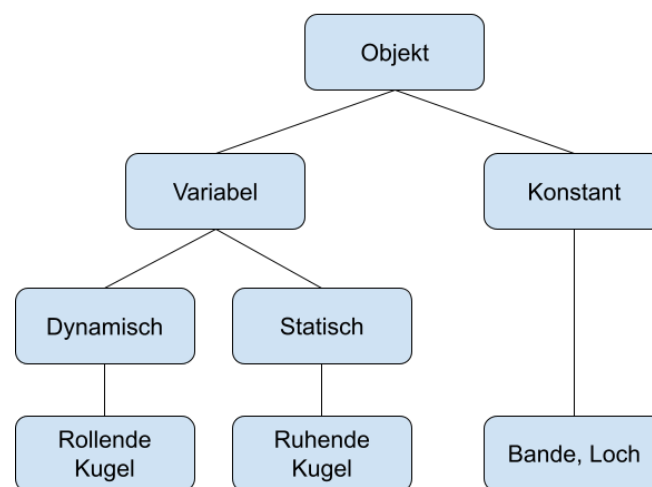


Abbildung 4.6: Typen von Objekten

4.4.3.1 Ereignisse und ihre Repräsentation

Das Ziel ist der Aufbau eines graphenähnlichen Konstrukts, welches aus Layern besteht und Zustandsübergänge variabler Objekte durch Knoten repräsentiert. Einige dieser Knoten treten bei Ereignissen wie einer Kollision oder wenn eine Kugel in ein Loch rollt auf. Andere Knoten dienen der reinen Abbildung des variablen Objekts innerhalb des Layers. Nachfolgend werden die Knoten definiert.



Energy-Input-Node: Dieser Node beschreibt das Auftreten eines Energie-Inputs von Aussen. Beispielsweise wenn die weisse Kugel mit dem Queue gestossen wird.



Energy-Transfer-Node (Collision-Node): Dieser Node beschreibt die Übergabe von Energie auf die beteiligten Objekte. Ein Beispiel ist die Kollision zweier Kugeln, wobei diese Energie austauschen.

Der Energy-Transfer-Node wird in drei Schichten aufgeteilt. Bei der Input-Schicht wird die Kantenfunktion (siehe S. 17) angewendet. Diese berücksichtigt den Energieverlust bis zum Auftreten des Ereignisses. Die mittlere Schicht beschreibt die Übergabefunktion der beteiligten Objekte. Die dritte Schicht repräsentiert das Resultat, also den Status der Objekte nach der Energieübergabe.

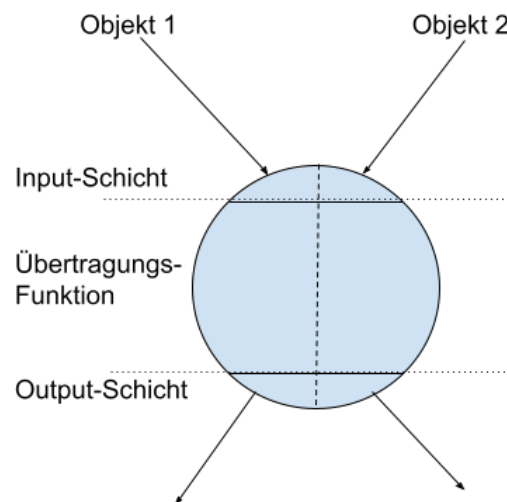


Abbildung 4.7: Der Energy-Transfer-Node



No-Energy-Node: Dieser Node wird eingesetzt, sobald der Energiewert eines variablen Objekts auf 0 sinkt und somit auch den Übergang von dynamisch zu statisch repräsentiert.



Out-Of-System-Node: Dieser Node wird eingesetzt, sobald ein variables Objekt das System verlässt, indem beispielsweise eine Kugel in ein Loch rollt.



Cutting-Node: Dieser Node ist ein spezieller Energy-Transfer-Node. Er wird bei variablen Objekten eingesetzt, die nicht an einem Ereignis beteiligt sind, wenn ein solches auftritt. Zum Ereigniszeitpunkt wird ein neuer Layer (siehe S. 17) geschaffen, welcher die Objekte, die am Ereignis beteiligt sind, in einem Energy-Transfer-Node festhält. Weiterhin wird der Status aller dynamischen Objekte ebenfalls zu diesem Zeitpunkt festgehalten. Der Cutting-Node beschränkt sich auf ein Input- sowie Output-Objekt und beinhaltet als Energieübertragungsfunktion die Identitätsfunktion.

4.4.3.2 Kantenfunktion

Die Kantenfunktion beschreibt eine Energieabnahme über die Zeit oder den Weg.

4.4.3.3 Layer

Ein Layer beinhaltet sämtliche Veränderungen der variablen Objekte und beschreibt den Status aller Objekte zu diesem Zeitpunkt. Sobald ein Ereignis auftritt wird ein neuer Layer im Modell eingefügt.

4.4.3.4 Beispiel eines Graphen

Das beschriebene Datenmodell kann als Graph¹ visualisiert werden. Als Beispiel wird die Idee eines Billardstosses in Abbildung 4.8 hinzugezogen. Auf dem Tisch liegt eine weisse wie auch zwei rote Kugeln. Die zweite rote Kugel wird an keiner Interaktion beteiligt sein, weswegen sie ihren Zustand nie verändert. Es ist ersichtlich, dass bei jedem Ereignis für diese Kugel ein „Out-of-energy-Node“ eingefügt wird. Das erste Event beschreibt die Kollision des dynamischen Objekts „weisse Kugel“ sowie des statischen Objekts „rote Kugel“. Danach verliert die weisse Kugel sämtliche Energie und wechselt in den Status „Out-of-energy“. Da die rote Kugel zu diesem Zeitpunkt dynamisch ist, wird für sie ein „Cutting-Node“ eingefügt. Zuletzt rollt die rote Kugel in das Loch, für sie wird ein „Out-of-system-Node“ erstellt und das System hat sämtliche Energie verloren. Ein Endzustand wurde erreicht, da alle Kugeln nun statisch sind.

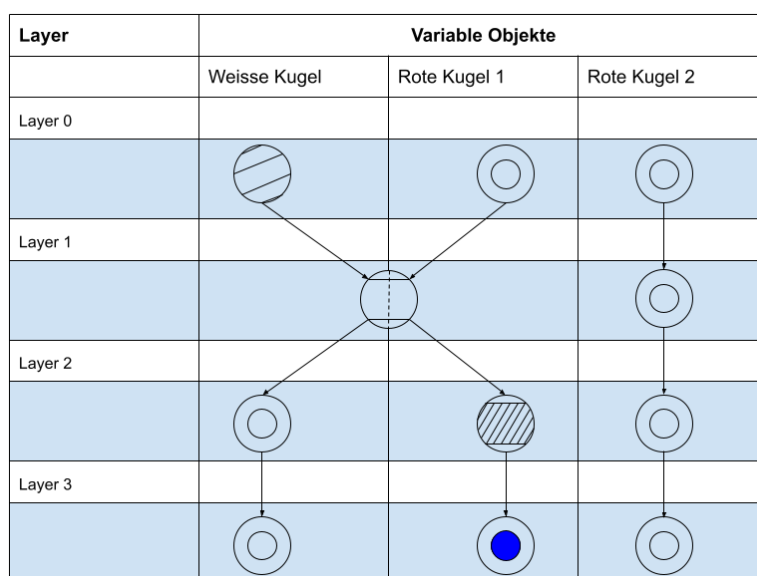


Abbildung 4.8: Beispiel für ein Resultat des Algorithmus 2

In Algorithmus 2 wird die Grundidee erläutert. Als Input für die Funktion „simulate“ dient der erste Layer, welcher mehrere variable Objekte beinhalten kann, wobei mindestens ein Objekt dynamisch (energiereich) sein sollte. Dieser Layer wird direkt dem erzeugten System hinzugefügt und dieses wird solange bearbeitet, bis alle variablen Objekte statisch sind (das System hat keine Energie mehr). In jedem Schleifendurchlauf wird das nächste Event berechnet. Die Events können diverser Natur sein. Es werden Kollisionen mit dynamischen, statischen wie auch konstanten Objekten geprüft. Bei der Kollision mit konstanten Objekten können die Ereignisse „Energy transfer“ oder „Out of System“ auftreten. Weiterhin wird geprüft, ob ein dynamisches Objekt seine Energie durch die Kantenfunktion verliert. Aufgrund eines Ereignisses wird zunächst ein neuer Layer generiert, dann für die am Ereignis beteiligten Objekte den entsprechenden Node und zuletzt für die anderen variablen Objekte entweder

¹Aus Effizienzgründen wird auf einen Graphen in der Implementation verzichtet. Das Modell setzt sich aus verschiedenen Layern zusammen, bei welchen die Informationen der Events für jedes variable Objekt separat und mit Zugriffszeit $O(1)$ abgelegt werden.

ein „Cutting-Node“ oder ein „Out-of-energy-Node“ eingefügt.

```
Function simulate(start: Layer, constantObjects: list)  $\rightarrow$  System
| system  $\leftarrow$  System()
| system  $\leftarrow$  appendLayer(system, start)
| while ! system.isStatic() do
| | nextEvent  $\leftarrow$  nextEvent(system.lastLayer(), constantObjects)
| | layer  $\leftarrow$  atMoment(nextEvent)
| | system  $\leftarrow$  appendLayer(system, start)
| end
| return system

Function nextEvent(layer: Layer, constantObjects: list)  $\rightarrow$  Node
| nextEvent: Node  $\leftarrow$  none
| for object in layer.dynamicObjects() do
| | nextEvent  $\leftarrow$  min(nextEvent, outOfEnergy(object))
| | nextEvent  $\leftarrow$  min(nextEvent, collision(object, layer.dynamicObjects()))
| | nextEvent  $\leftarrow$  min(nextEvent, collision(object, layer.staticObjects()))
| | nextEvent  $\leftarrow$  min(nextEvent, collision(object, constantObjects))
| end
| return nextEvent
```

Algorithm 2: Algorithmus zum Aufbau eines physikalischen Systems

4.4.4 Simulation

Sobald ein möglicher Lösungskandidat anhand der in Abschnitt 4.4.1 beschriebenen Suche gefunden und dessen Initialgeschwindigkeit nach Abschnitt 4.4.2 berechnet wurde, wird eine Simulation durchgeführt, um die Lösung definitiv zu bestätigen. Durch das Anwenden verschiedener Anfangsgeschwindigkeiten der weissen Kugel können in diesem Schritt mehrere Situationen evaluiert werden.

Die Simulation wird durch die Definition eines physikalischen Systems wie in Kapitel 4.4.3 durchgeführt. Hierbei gelten die Zuordnungen wie sie nachfolgend beschrieben werden.

Ereignisse

Energy-Input-Node Wird modelliert über die Eingabe der Energie der weissen Kugel. Ein spezifischer Node zur Modellierung wird nicht implementiert, es wird der Energy-Transfer-Node verwendet, wobei nur der Output-Wert relevant ist.

Energy-Transfer-Node Tritt bei der Kollision zwischen zweier Kugeln oder einer Kugel mit der Bande auf.

No-Energy-Node Tritt auf, wenn eine Kugel vom dynamischen in den statischen Zustand wechselt (ausrollt). In jedem Layer, wo eine Kugel statisch ist, wird sie durch diesen Node modelliert.

Out-of-System-Node Sobald eine Kugel mit dem Zielkreis kollidiert, tritt dieses Ereignis auf. Dem System wird die Energie entzogen und die Kugel ist nicht mehr verfügbar.

Kantenfunktion Die Kantenfunktion zwischen den Übergängen innerhalb des Layers bildet der Reibungsverlust der Kugel über eine bestimmte Zeit oder einen bestimmten Ort.

Dynamische/Statische Objekte Im Billiard gibt es nur die Kugeln als statische und/oder dynamische Objekte.

Konstante Objekte Die konstanten Objekte bilden die Banden wie auch die Ziele.

Es wird ungefähr der Pseudoalgorithmus wie in 2 angewendet, optimal auf das Problem „Billiard“ abgestimmt. Es folgen die physikalischen Berechnungen zur Durchführung der Simulation.

4.4.4.1 Reibungsverlust über Strecke

TODO: T-5 Reibungsverlust über eine Strecke

4.4.4.2 Ereignis Out-Of-Energy

TODO: T-5 Auftrittszeitpunkt Out-Of-Energy über Reibungsverlust

4.4.4.3 Ereignis Energy-Transfer über Kugelkollision

TODO: T-5 Kollision mit Kugeln (statisch/dynamisch) berechnen

4.4.4.4 Ereignis Energy-Transfer über Bandenkollision

TODO: T-5 Kollision mit Banden berechnen.

4.4.4.5 Ereignis Out-Of-System

TODO: T-5 Kollision mit Zielkreis berechnen.

4.4.5 Animation

TODO: einleitung etc.

Ein Layer wird grundsätzlich in zwei Bereiche aufgeteilt. Es gibt den Inputbereich, welcher den Status eines variablen Objektes vor der Energieübertragung, sowie den Outputbereich, welcher den Status eines variablen Objektes nach der Energieübertragung beschreibt.

Ein Layer definiert immer deren zwei Keyframes, sofern es sich nicht um den Input- oder Output-Layer handelt. Um den Zustand des Systems zu einem beliebigen Zeitpunkt zu berechnen, kann jeweils zwischen den KeyFrames n und $n + 1$ zweier Layer interpoliert werden.

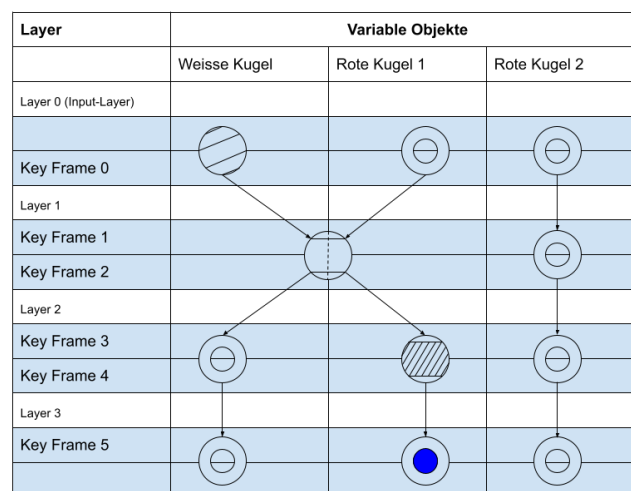


Abbildung 4.9: Animation keyframes

4.5 Berechnungsprozess

Die Berechnung eines optimalen Stosses wird aufgrund der vielen Möglichkeiten sehr zeitintensiv, weswegen die expandierten Teilschritte parallel gerechnet werden.

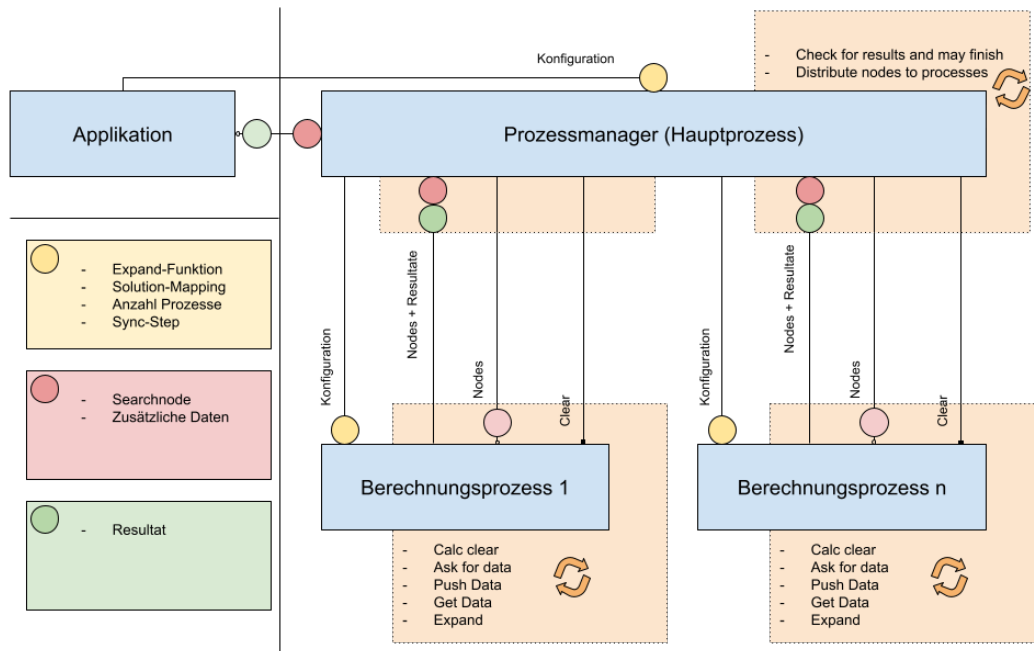


Abbildung 4.10: Berechnungsprozess

Abbildung 4.10 erläutert die Durchführung dieser Expansionsschritte. Die Applikation auf der linken Seite liefert die konkreten Aufgaben, welche von einem Pool an Berechnungsprozessen bearbeitet werden sollen. Sie beinhaltet die Logik, um eine solche Aufgabe zu lösen, da die Berechnungsprozesse selbst nicht über dieses Wissen verfügen. Der Pool von Berechnungsprozessen wird vom Prozessmanager verwaltet und dieser ist die zentrale Ansprechperson der Applikation.

Datenaustausche sind über gefärbte Kreise repräsentiert. Handelt es sich um einen asynchronen Datenaustausch, dann ist der entsprechende Kreis heller eingefärbt und die Verbindungslinie ist durch einen nicht ausgefüllten Punkt gekennzeichnet. Es werden die Farben Rot den zu bearbeitenden Daten, Gelb den Konfigurationsdaten und Grün den Resultaten zugeordnet. Signale sind durch ein ausgefülltes Quadrat an der Verbindungslinie gekennzeichnet. Das Signal oder die Daten sind jeweils beim Empfänger angegeben. Repetitive Aufgaben sind durch ein hinterlegtes oranges Rechteck markiert.

Bei der Erstellung des Prozessmanagers werden alle Konfigurationen mitgeliefert. Der Prozessmanager erzeugt während seiner Instanziierung die Prozesse, welche ebenfalls direkt konfiguriert werden. Die Prozesse laufen nun im Hintergrund und warten auf Daten.

Erhält die Applikation eine Suchanfrage, ruft sie in einem ersten Schritt den Prozessmanager auf. Dieser nimmt noch zu berechnende Daten entgegen. Im Fall von Billard sind dies die Root-Nodes des Suchbaums. Jeder Root-Node repräsentiert ein Loch. Der Prozessmanager gibt in dem Fall eine Datenstruktur zurück, welche es erlaubt, in Zukunft auf die Anfrage zu antworten. In dieser Antwort werden die Resultate geliefert. Der Hauptprozess prüft regelmässig, ob die Bearbeitung der gestellten Aufgaben beendet werden soll. Dies ist der Fall, wenn entweder genügend Lösungen gefunden wurden oder die maximal zur Verfügung stehende Zeit abgelaufen ist. Sollte die Berechnung nicht abgebrochen werden, dann werden die offenen Anfragen der Berechnungsprozesse beantwortet. Der Prozessmanager verteilt alle ihm gemeldeten noch zu bearbeitenden Daten an die einzelnen Berechnungsprozesse.

In Abbildung 4.11 wird das Ziel verdeutlicht, welches der Prozessmanager bei einer Verteilung dieser Daten verfolgt. So sollen die jeweils n besten Kandidaten, wobei n für die Anzahl der Berechnungsprozesse steht, auf die verschiedenen Berechnungsprozesse verteilt werden. Danach erhalten die Berechnungsprozesse die nächstbesten n -Kandidaten. Dadurch wird sichergestellt, dass immer an den vielversprechendsten Arbeitspaketen weitergemacht wird.

Der Algorithmus 3 zeigt, wie die offenen Arbeitspakete auf die Berechnungsprozesse verteilt werden, damit das Ziel in Abbildung 4.11 erreicht werden kann. So werden zuerst alle Antworten auf die offenen Anfragen erstellt. Dies geschieht in einer „While-Loop“, welche solange läuft, wie es noch offene Arbeitspakete hat. In jedem Schritt der Schleife

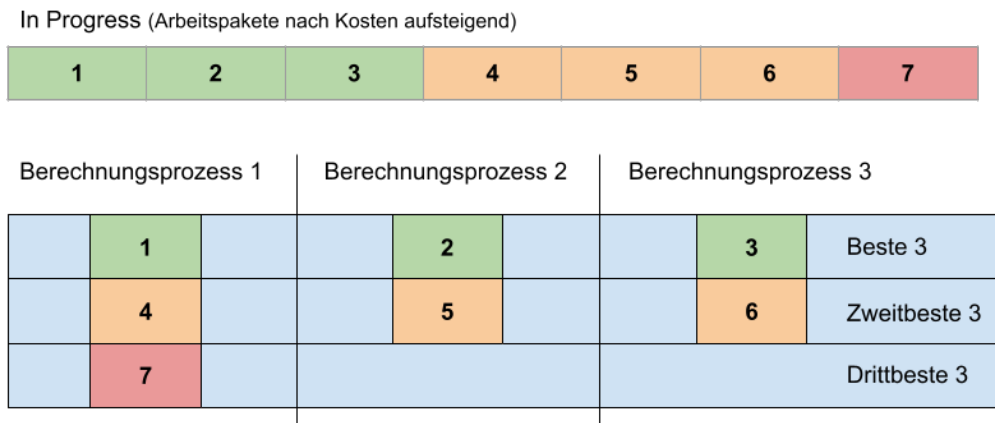


Abbildung 4.11: Berechnungsprozessverteilung

fe wird ein Arbeitspaket einer Antwort für eine Anfrage hinzugefügt. Am Ende der Schleife wird eine Index-Variable hochgezählt. Sollte diese die Anzahl der offenen Anfragen erreichen, beginnt sie wieder bei 0. Dies ist z.B. in Abbildung 4.11 beim Wechsel von Arbeitspakete 3 zu 4 ersichtlich. Sobald keine offenen Arbeitspakete mehr existieren, werden sämtliche Anfragen beantwortet, zu welchen Antworten erstellt werden konnten. Wenn z.B. mehr Anfragen als Arbeitspakete existieren, bleiben diese erhalten und werden im nächsten Synchronisationszyklus beantwortet².

Function `distribute(requests: List[future], inProgress: Queue[Node]) → List[future]`

```

answers: List[List[Node]] ← list()
forRequest ← 0
while ! inProgress.empty() do
  if answers[forRequest] is null then
    | answers[forRequest] = list()
  end
  node ← pop(inProgress)
  answers[forRequest] ← append(node, answers[forRequest])
  forRequest ← (forRequest + 1) % size(requests)
end
while ! answers.empty() do
  request ← pop(requests)
  fulfill(request, pop(answers))
end
return requests

```

Algorithm 3: Algorithmus zur Durchführung eines Expansionsschritts im Berechnungsprozess

In einem Zyklus eines Berechnungsprozesses wird zuerst geprüft, ob eine Berechnung abgebrochen werden soll. Dies ist der Fall, wenn der Prozessmanager ein „Clear-Signal“ gesendet hat. Dieses Signal tritt auf, wenn eine neue Berechnung gestartet oder wenn eine aktuell Laufende erfolgreich beendet oder abgebrochen wird.

Der Berechnungsprozess stellt in einem nächsten Schritt eine Datenanfragen an den Prozessmanager, sollte keine offene oder noch nicht bearbeitete Anfrage existieren. Danach erfolgt die Prüfung, ob die Zeit zwischen einem Synchronisationsschritt abgelaufen ist. In dem Fall liefert der Berechnungsprozess seine besten weiterzuführenden Berechnungen wie auch Resultate dem Prozessmanager. Er wird demnach für eine Weile an weniger erfolgsversprechenden Resultaten weiterarbeiten. In einem weiteren Schritt wird geprüft, ob eine Datenanfrage bereits beantwortet wurde. Ist dies der Fall, so werden die Daten den zu bearbeitenden Daten des Berechnungsprozesses hinzugefügt. Zuletzt erfolgt der Expansionsschritt, wobei der erfolgsversprechendste Kandidat expandiert wird. Der Algorithmus 4 zeigt, wie dies in etwa funktionieren kann. Als Eingabe dient eine priorisierte Queue von offenen Arbeitspaketen sowie die Implementation der „Expand-Funktion“. Wenn es offene Arbeitspakete gibt, dann wird das Beste genommen und über die Implementation der „Expand-Funktion“ expandiert. Das Resultat dieses Schrittes wird danach entweder den Lösungen oder den offenen Arbeitspaketen hinzugefügt. Am Schluss werden die offenen

²Man stelle sich bei der Abbildung 4.11 nur zwei offene Arbeitspakete vor. Demnach würde Paket 1 auf den Berechnungsprozess 1 und Paket 2 auf den Berechnungsprozess 2 verteilt werden. Die Anfrage des Berechnungsprozesses 3 bliebe offen.

Arbeitspakete wie auch die Lösungen zurückgegeben.

```

Function expand(inProgress: Queue[Node], expandImpl: functional)  $\longrightarrow$  Queue[Node], Queue[Node]
  solutions  $\leftarrow$  priority_queue()
  if !inProgress.empty() then
    node  $\leftarrow$  pop(inProgress)
    expandedNodes  $\leftarrow$  expandImpl(node)
    for expandedNode in expandedNodes do
      if expandedNode.isSolution() then
        | solutions  $\leftarrow$  append(expandedNode, solutions)
      end
      else
        | inProgress  $\leftarrow$  append(expandedNode, inProgress)
      end
    end
  end
  return inProgress, solutions

```

Algorithm 4: Algorithmus zur Durchführung eines Expansionsschritts im Berechnungsprozess

Die Synchronisation zwischen den Prozessen wird in Abbildung 4.12 erläutert. Sie findet nach einer konfigurierten Zeit k statt. Diese wird in Millisekunden [ms] angegeben. Da die Synchronisation ein exklusives Verwenden einer geteilten Ressource erfordert, um die erledigte Arbeit dem Prozessmanager mitzuteilen, erhalten alle laufenden Prozesse (dies umfasst den Hauptprozess des Prozessmanagers wie auch alle Berechnungsprozesse) ein Zeitfenster zugeteilt, welches nach k Millisekunden startet. Danach kann jeder Prozess nach $k + i \cdot n$, wobei i für die Prozessid beginnend bei 0 und n für das Zeitfenster einer Synchronisation steht, mit der Synchronisation beginnen. Es wird so verhindert, dass zu viele Prozesse auf die Ressource des Prozessmanagers warten müssen und untätig bleiben. Ein weiterer Vorteil dabei ist die Tatsache, dass in der Zeit der Synchronisation einige Prozesse Zeit erhalten, um eher schlechter bewertete Kandidaten weiterzuverfolgen. Dadurch ist es möglich, dass ein sehr gutes Ergebnis gefunden werden kann, obwohl dessen Kandidat anfänglich als schlecht beurteilt wurde.

| Prozess\Zeit [ms] | k | k + 0n | k + n | k + 2n | 2k | 2k + 0n | 2k + n | 2k + 2n |
|-------------------|---|--------|-------|--------|----|---------|--------|---------|
| Hauptprozess | | | | | | | | |
| Berechnungspr. 1 | | | | | | | | |
| Berechnungspr. 2 | | | | | | | | |

Abbildung 4.12: Berechnungsprozesssynchronisation

Die Abbildung 4.12 zeigt den Hauptprozess wie auch zwei Berechnungsprozesse, welche eine Synchronisation in einem Intervall von k Millisekunden durchführen. Grüne Spalten stehen hierbei für die Zeit, welche für die Berechnung einer Lösung verwendet wird, also als produktiv bezeichnet werden kann. Rote Spalten hingegen signalisieren den unproduktiven Overhead, der bei der Synchronisation entsteht. Der Prozessmanager macht nebst seinem Synchronisationsfenster nichts, weswegen seine Spalten grau markiert sind. Eine mögliche Verbesserung wäre die Auslastung des Prozessmanagers mit zusätzlicher Berechnungsarbeit, wie sie die Berechnungsprozesse durchführen.

5 Resultate

Dieses Kapitel beinhaltet die Aufführung aller Ergebnisse der Arbeit. Dies betrifft insbesondere die Fortsetzung der Genauigkeitsanalyse der Kugeldetektion aus der Vorarbeit[Luk21b], die Genauigkeitsanalyse der Klassifikation wie auch eine Aufstellung einiger Spielstände, deren Suchresultate und verwendete Berechnungszeiten für die einfache direkte wie auch erweiterte Suche.

5.1 Klassifikation

6 Weitere Arbeiten

TODO: Weitere Arbeiten

7 Fazit

TODO: Fazit

Abbildungsverzeichnis

| | | |
|------|---|----|
| 4.1 | Billard-AI-Cycle | 9 |
| 4.2 | Risikoanalyse | 11 |
| 4.3 | Kandidatensuche 1 | 12 |
| 4.4 | Kandidatensuche 2 | 13 |
| 4.5 | Kandidatensuche 3 | 13 |
| 4.6 | Typen von Objekten | 15 |
| 4.7 | Der Energy-Transfer-Node | 16 |
| 4.8 | Beispiel für ein Resultat des Algorithmus 2 | 17 |
| 4.9 | Animation keyframes | 19 |
| 4.10 | Berechnungsprozess | 20 |
| 4.11 | Berechnungsprozessverteilung | 21 |
| 4.12 | Berechnungsprozesssynchronisation | 22 |

Tabellenverzeichnis

| | | |
|-----|---------|----|
| 3.1 | Ziele | 6 |
| 4.1 | Risiken | 10 |

Literatur

[Luk21a] Luca Ritz Lukas Seglias. “Billiard-AI”. In: (2021), S. 5–6.

[Luk21b] Luca Ritz Lukas Seglias. “Billiard-AI”. In: (2021), S. 21–23.



Erklärung der Diplomandinnen und Diplomanden *Déclaration des diplômé-e-s*

Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/*Nom*, Vorname/*Prénom*

Datum/*Date*

Unterschrift/*Signature*

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.



Erklärung der Diplomandinnen und Diplomanden *Déclaration des diplômé-e-s*

Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/*Nom*, Vorname/*Prénom*

Datum/*Date*

Unterschrift/*Signature*

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.

8 Anhang

TODO: Anhang