Hochschule Neu-Ulm
University of Applied Sciences

Bachelorarbeit
im Bachelorstudiengang
Game-Produktion und Management
an der Hochschule für angewandte Wissenschaften Neu-Ulm

Identifizierung von relevanten Algorithmen in der Videospielentwicklung und wie diese Anwendung finden

Erstkorrektor: Prof. Dr. Stefan Faußer
Zweitkorrektor: Sebastian de Andrade

Verfasser: Jonathan Buss (Matrikel-Nr.: 303280)

Thema erhalten: 30.04.2025
Arbeit abgegeben: 01.09.2025

Im Rahmen dieser Abschlussarbeit wird das generische Maskulinum verwendet.

Abstract

Diese Bachelorarbeit untersucht, welche Algorithmen in der Videospielentwicklung derzeit am relevantesten sind und wie diese in Game Engines Anwendung finden.
Die Arbeit soll Anfängern im Bereich der Spielalgorithmen einen zielgerichteten Einstieg ermöglichen und Lehrenden, die bislang wenig Berührungspunkte mit Spielen haben, einen Bezug zwischen Algorithmen und ihrer konkreten Anwendung in der Spieleentwicklung vermitteln. Ziel ist es, eine Übersicht relevanter Algorithmen zu erstellen und deren praktische Umsetzung anhand von Beispielen zu analysieren. Bestehende Literatur, behandelt Algorithmen überwiegend theoretisch, bieten jedoch keine systematische Einordnung nach bestehenden Genres oder eine praxisnahe vergleichende Darstellung der Implementierung in populären modernen Game Engines.

Zur Schließung der Forschungslücke werden relevante Algorithmen zunächst mittels einer quantitativen Literaturanalyse und einer qualitativen Expertenanalyse ermittelt. Anschließend werden ausgewählte Algorithmen exemplarisch in führende Game Engines implementiert, um ihre praktische Anwendung in der realen Spieleentwicklung zu analysieren.

Die Ergebnisse zeigen, dass insbesondere Movement-, Decision-Making- und Pathfinding-Algorithmen eine zentrale Rolle in der Spieleentwicklung einnehmen, während Learning-Algorithmen zunehmend an Relevanz gewinnen. Ein signifikanter Zusammenhang zwischen Genres und Algorithmen konnte nicht nachgewiesen werden, was auf eine genreübergreifende Nutzbarkeit hinweist. Die Experteninterviews verdeutlichen, dass viele Verfahren bereits in Engines integriert oder über Packages verfügbar sind, deren Bedeutung sich vor allem durch die jeweilige Implementierungstiefe bestimmt. Damit leistet die Arbeit einen Beitrag zur Verbindung von theoretischer Klassifikation und praktischer Anwendung, zeigt aber auch, dass erweiterte Datensätze sowie die Berücksichtigung neuer Technologien für eine noch umfassendere Bewertung erforderlich sind.

Key words: Algorithms, Video Game Industry, Game Development, Implementation, Genre

Inhaltsverzeichnis

Abstract ..... **II**
Abbildungsverzeichnis ..... **V**
Tabellenverzeichnis ..... **VI**
Abkürzungsverzeichnis ..... **VII**
Glossar ..... **VIII**
1 Einleitung **1**
1.1 Überblick *1*
1.2 Forschungsstand *2*
1.3 Forschungsfrage und Zielsetzung *3*
1.4 Forschungsmethodik *4*
1.4.1 Quantitativer Teil 5
1.4.2 Qualitativer Teil 7
2 Grundlagen **8**
2.1 Definition von Relevanz *8*
2.2 Definition von Algorithmus *9*
3 Quantitative Recherche **11**
3.1 Vorbereitung der Daten *11*
3.2 Aufbereitung der Daten *13*
3.3 Beobachtung der Genres *16*
3.4 Beobachtung der Algorithmen *18*
3.5 Zusammenhang zwischen Genre und Algorithmen *21*
3.6 Ergebnis der Quantitativen Recherche *23*
4 Qualitative Experteninterviews **24**
4.1 Welche weiteren Kategorien berücksichtigt werden sollten *25*
4.2 Welche Algorithmen essenziell sind *27*
4.3 Welche Algorithmen wurden bereits benutzt wurden *29*
4.4 Wann und wo wurden diese Algorithmen benutzt wurden *31*
4.5 Ergebnis der Experteninteviews *33*
5 Überprüfung Und Bewertung der Rechercheergebnisse **35**
5.1 Validität und Reliabilität der Ergebnisse *35*

5.2 Methodische Einschränkungen und Verzerrungen 36

5.3 Einordnung der quantitativen Ergebnisse mithilfe qualitativer Interviews 37

5.4 Ergänzende Experteninterviews auf der Gamescom 38

6 Ansätze für weiterführende Arbeiten 40

7 Fazit 41

Literaturverzeichnis IX

Übersicht verwendeter Hilfsmittel XII

Anhang XIII

Abbildungsverzeichnis

Abbildung 1: Darstellung eines Explanativen Mixed-Methods-Research Designs (Ngulube, 2021, S. 29) 4

Abbildung 2: Trendentwicklung der Genres aller nominierten Spiele bei den Game Awards 2014–2024 (eigene Darstellung) 16

Abbildung 3: Anzahl Kategorien pro Spiel, welche mindestens ein Algorithmus vorweisen (eigene Darstellung) 18

Abbildung 4: Prozentuale Trendentwicklung der AI-Kategorien in den Jahren 2014-2024 (eigene Darstellung) 19

Abbildung 5: Kontingenztafel zwischen Genre und AI-Kategorie (eigene Darstellung) 21

Tabellenverzeichnis

Tabelle 1: Gefundenen Stichwörter der kuratierten Liste, eingeteilt in die AI-Kategorien nach Millington 15

Tabelle 2: Kontingenztafel zwischen Genre und AI-Kategorie mit der Anzahl verschiedener Spiele als Wert (eigene Tabelle) 21

Abkürzungsverzeichnis

AAA *High budged video game productions by major studios with sophisticated team, generating large revenues* (Bossom & Dunning, 2015, S. 200)

AI *Artificial Intelligence*

CSV (Comma-Separated Values) Ein Dateiformat zur Speicherung und zum Austausch tabellarischer Daten, bei dem einzelne Werte durch Trennzeichen getrennt werden.

GDD *German Dev Days*

Indie *Independent productions by small teams, often without publisher and external support in financing* (Bossom & Dunning, 2015, S. 201)

KI *Künstliche Intelligenz*

LLM *Large Language Models (kurz: LLM und auf Deutsch: Große Sprachmodelle) sind leistungsstarke Modelle, die darauf ausgelegt sind, menschliche Sprache zu verstehen und zu generieren.* (Kelbert, Siebert, & Jöckl, 2023, Abschnitt 1)

NPCs *Nicht-Spieler-Charaktere*

Glossar

Pathfinding (deutsch: Wegsuche oder Pfadsuche)
Die Berechnung eines effizienten oder optimalen Weges von einem Startpunkt zu einem Zielpunkt.

Decision Making (deutsch: Entscheidungsfindung)
Der Prozess, bei dem aus mehreren Handlungsoptionen eine Auswahl getroffen wird.

Content (deutsch: Inhalt, In Bezug auf Videospiele)
Inhalte, die das Spielerlebnis ausmachen. Dazu gehören sichtbare Elemente, wie Figuren oder Umgebungen und unsichtbare Strukturen, wie Dialoge oder Spielmechaniken.

Game Engine (auch nur Engine genannt)
Eine Software, die grundlegende Funktionen für die Entwicklung von Videospielen bereitstellt. Dabei wird spielspezifischer Content nicht bereitgestellt.

Game AI (deutsch: Spiel-KI)
Die künstliche Intelligenz, die sich in Spielen befindet oder zur Erstellung von Spielen genutzt wird. In Gegensatz zur allgemeiner KI liegt der Fokus auf der Simulation von glaubwürdigem, spaßförderndem Verhalten und nicht unbedingt auf echter „Intelligenz“.

Pseudo-Code
“[…] an imaginary programming language that cuts out any implementation details particular to any real programming language.” (Millington, 2019, S. 15)

Co-op (Abkürzung für „Cooperative Play“; Deutsch: kooperatives Spielen)
Bezeichnet einen Spielmodus in Videospielen, bei dem mehrere Spieler gemeinsam statt gegeneinander spielen.

Movement (deutsch: Bewegung)
Die Bewegung von Spieler oder NPCs ein einem Videospiel.

Tactical and Strategic AI
Mechanismen, die über rein lokale Entscheidungen hinausgehen und komplexe taktische oder langfristige strategische Handlungen ermöglichen, wie sie etwa in Echtzeit- oder Rundenstrategiespielen benötigt werden.

Learning AI (deutsch: lernende KI)
Ansätze, bei denen das Verhalten von NPCs oder Spielsystemen durch Erfahrungswerte oder Anpassungsprozesse optimiert wird.

Procedural Content Generation (deutsch: prozedurale Content Generierung)
Algorithmen zur automatischen Erzeugung von Content

Board Games
Spezielle Verfahren, die klassische Brettspielmechanismen simulieren, etwa Minimax-Funktionen in Spielen mit klar definierter Zugfolge.

Hitbox
Ein unsichtbarer Bereich, um ein Spielelement, der zur Erkennung von Treffern und Kolisionen dient.

Bubblesort
“[…] is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.” (Geeks for Geeks Sanchhaya Education Private Limited, 2025, Abschnitt 1)

1 Einleitung

1.1 Überblick

Algorithmen sind wesentlicher Bestandteil von Videospielen. Sie bestimmen technische Prozesse wie das Pathfinding, Physiksimulationen oder die prozedurale Generierung von Content und prägen auch maßgeblich das Spielerlebnis durch Decision Making, künstliche Intelligenz und Optimierungsverfahren. Obwohl grundlegende algorithmische Prinzipien wie Suchverfahren, Grafenalgorithmen oder heuristische Ansätze seit Jahrzenten existieren, findet ihre konkrete Umsetzung in modernen Game Engines und für unterschiedliche Spielgenres in sehr variierter Form statt.

Die Forschungslage zu Spielalgorithmen ist umfangreich, fokussiert sich jedoch häufig auf theoretische Darstellungen einzelner Verfahren oder auf hochspezialisierte Teilbereiche. Eine systematische Einordung relevanter Algorithmen im Kontext der aktuellen Videospielentwicklung in Bezug auf Genres und ihrer praktischen Anwendung in Game Engines existiert nicht. Es ergibt sich somit eine Forschungslücke zwischen Theorie und praktischer Anwendung. Hier setzt die vorliegende Arbeit an.

Ziel der Arbeit ist es, zu klären, welche Algorithmen in der aktuellen Spieleentwicklung besonders relevant sind. Darüber hinaus wird untersucht, ob ein Zusammenhang zwischen Genres und Algorithmen besteht und wie diese in der Industrie konkret angewandt werden. Grundlage bildet eine quantitative Literaturanalyse zur Identifikation relevanter Verfahren, ergänzt durch qualitative Experteninterviews zur Validierung und Erweiterung der Ergebnisse. Auf diese Weise wird die bestehende Lücke zwischen abstrakter Theorie und praktischer Anwendung geschlossen.

1.2 Forschungsstand

Es gibt Standardwerke, die die Game AI auflisten und diese thematisieren. Die prominentesten Werke dazu sind „AI for Games“[1] und „Artificial Intelligence in Games“[2] Diese Werke betrachten verschiedenste Algorithmen, die in der Spieleentwicklung verwendet werden. Dabei wird die Theorie der Prozessabläufe detailliert dargestellt und Implementationen in Pseudo-Code vorgeführt.

Die Sammelband Buchreihen „AI Game Programming Wisdom“[3] und „Game AI Pro“[4] von Roberts liefern einen angewandten Blick. In Jedem Sammelband kommen über mehr als 40 Experten zusammen und jeder liefert einen konkreten Blick auf die Implementierung von bestimmten Algorithmen in echten Projekten. Besonders die letzteren vier Sammelbände bieten eine zugängliche Informationsquelle, da diese kostenlos im Internet zu finden sind.[5]

In den Standardwerken kommen allerdings nur selten Projekte vor, die mit den am meisten genutzten Game Engines erstellt wurden.

Für diese Arbeit liegt ein Exemplar von „AI for Games“[6] vor und wird als Referenzwerk beigezogen, da dieses Werk die umfassendste Ansammlung an Algorithmen bietet. In den letzten zwei Kapiteln „Designing Game AI“[7] und „AI-Based Game Genres[8]“ teilt Millington die Algorithmen in die kommerziell wichtigsten[9] Genre ein. Diese sind „Shooters“, „Driving“, „Real-Time Strategy“, „Sports“, „Turn-Based Strategy Games“ in Kapitel 14. In Kapitel 15 nutzt Millington eigens kategorisierte Genre, die auf die spezielle Nutzung von bestimmten

---

[1] Millington, 2019

[2] Roberts, 2022

[3] vgl. Rabin, AI Game Programming Wisdom, 2002; Rabin, AI Game Programming Wisdom 2, 2003; Rabin, AI Game Programming Wisdom 3, 2006; Rabin, AI Game Programming Wisdom 4, 2008

[4] vgl. Rabin, Game AI Pro, 2013; Rabin, Game AI Pro 2, 2015; Rabin, Game AI Pro 3, 2017; Rabin, Game AI Pro 4, 2021

[5] vgl. Rabin, Game AI Pro, o.J.

[6] Millington, 2019

[7] Millington, 2019, Kapitel 14

[8] Millington, 2019, Kapitel 15

[9] Dies geschieht nach Millingtons eigenem Ermessen. Es werden keine Quellen für diese Aussage angegeben (Millington, 2019, S. 936)

Algorithmen abzielen. Dazu gehören “Teaching Characters” und „Flocking and Herding Games“. Somit lässt Millington viele Genre aus, die kommerziell wichtig sind. Zukowski entdeckt zum Beispiel, dass in 2024 21% der Spiele mit mehr als 1000 Reviews auf Steam Co-op Spiele waren.[1] Nach Millington sind bestimmte Algorithmen für bestimmte Genre nützlich. Ob sich dies in der Praxis widerspiegelt und ob diese Einteilung sinngemäß ist, bleibt in der Fachliteratur unbeantwortet. Somit ergibt sich die Forschungslücke, dass die Einordnung von Algorithmen in bestimmte Spielgenres bisher nur unzureichend auf ihre praktische Relevanz und Umsetzung in modernen Game Engines überprüft wurde.

1.3 Forschungsfrage und Zielsetzung

Auf Grundlage der dargestellten Forschungslücke und des bisherigen Forschungstandes ergibt sich die Zentrale Forschungsfrage:

Was sind die relevantesten Algorithmen in der Videospielentwicklung?

Ziel dieser Frage ist es, durch eine systematische Analyse von Literatur und Experteninterviews, Algorithmen zu identifizieren, die in der Praxis am häufigsten verwendet werden. Um diese Frage weitreichend zu beantworten, werden zwei Teilfragen spezifischer untersucht.

1. **Was ist der Zusammenhang zwischen Algorithmen und Genres?**
   Diese Frage untersucht, ob die Zuordnung von Algorithmen zu Genres zutreffend ist. Dabei wird überprüft, ob sich in der Zuordnung von Genres relevanter Spiele zu genutzten Algorithmen ein Muster erkennen lässt, welches Zeigt, dass in bestimmten Genres bestimmte Arten von Algorithmen bevorzugt genutzt werden.

2. **Wie werden Algorithmen angewandt?**
   Ziel dieser Frage ist es über die Anwendung von Algorithmen in der Industrie aufzudecken. Diese Frage ist komplementär zur ersten Teilfrage. Mit ihr soll herausgefunden werden, wie die Algorithmen in der Entwicklung ausgewählt werden und wozu diese genutzt werden.

[1] vgl. Zukowski, 2024, Percentage of coop

1.4 Forschungsmethodik

Um die Forschungsfrage zu beantworten, wird ein Explanatives Mixed-Methods-Research Design angewandt. Es wird zunächst eine quantitative Recherche betrieben. Die quantitativen Daten werden gesammelt und analysiert, um erste Erkenntnisse zu gewinnen. Mit der quantitativen Recherche wird eine Beantwortung der ersten Teilfrage angestrebt. Das Ergebnis der quantitativen Recherche wird genutzt, um qualitative Experteninterviews aufzubauen. Die Experteninterviews dienen zu Beantwortung von Fragen, die sich aus der quantitativen Recherche ergeben, aber nicht weiter quantitativ begründet werden können, sondern einer qualitativen Antwort bedürfen. Diese sind übergreifend in der zweiten Teilfrage enthalten, welche eine qualitative Analyse erfordert. Somit werden die Teilfragen schrittweise beantwortet, während das Thema immer detaillierter betrachtet und die Antwort auf die Leitfrage immer weiter konkretisiert wird.

Abbildung 1: Darstellung eines Explanativen Mixed-Methods-Research Designs *(Ngulube, 2021, S. 29)*

Wie in Abschnitt 3 erklärt wird, wurden alle nominierten Spiele der Game Awards zusammengetragen. Diese umfassen 648 eindeutige Spiele. Eine automatische Auswertung von Artikeln der Game Developer Webseite liefert einen Datensatz von 2475 Artikeln. Mithilfe des Datensatzes von RAWG konnten sämtliche nominierten Spielen mindestens ein Genre plattformübergreifend zugeordnet werden. Dieser Datensatz bildet das Fundament der quantitativen Recherche. Ein Datensatz dieser Größe erlaubt die Identifizierung von Trends und bietet ausreichend Information für eine Identifizierung relevanter Algorithmen und der Erkennung von Mustern in der Zuweisung von Algorithmen zu Genres. Ergebnis der quantitativen Recherche ist eine Rangliste, die die relevantesten Algorithmen nach Häufigkeitsgrad der Nutzung zeigt.

Zur Erklärung der Erkenntnisse aus dem quantitativen Teil, dienen qualitative Experteninterviews. Die Experteninteviews zielen darauf ein tieferes Verständnis über die Implementierung von Algorithmen zu erlangen, um einen kausalen Zusammenhang zu bilden. Auf den German Dev Days 2025 (GDD) wurden dazu sechs Experteninterviews mit Programmierern aus der Industrie durchgeführt. Die Entwickler wurden zufällig ausgewählt und arbeiten zum Großteil in der Indie-Branche, es wurden allerdings auch Experten befragt, die an größeren Produktionen gearbeitet haben. Dabei wurde darauf geachtet keine Studenten zu interviewen, um Erfahrungen aus realer Industrie zu erlangen. Durch die qualitative Analyse der Experteninterviews wird eine finale Theorie aufgestellt.

1.4.1 Quantitativer Teil

Als primäre Quelle dienen Fachartikel aus der Game Developer online Fachzeitschrift. Diese bietet den größten Datensatz an Fachartikeln in der Industrie, die speziell für Spieleentwickler erstellt wurden. Game Developer enthält keine Werbeartikel oder Spekulationsartikel. Die Redaktionspolitik stellt sicher, dass die Artikel stets neutral sind.[1] Dies stellt die Legitimität sicher und erlaubt eine genaue Ermittlung der genutzten Algorithmen. Zur Durchsuchung der Game Developer Artikel wurde eine Stichliste erstellt. Diese Stichliste besteht zunächst aus dem Inhaltsverzeichnis von Millington.[2] Die nominierten Spiele sind plattformübergreifend. Steam[3] deckt nur PC-Spiele ab, weshalb die Steam Datenbank nicht zur Genreermittlung genutzt werden kann. Die RAWG[4] Datenbank hingegen ist plattformübergreifend und wurde genutzt, um Genre zu jedem nominierten Spiel zuzuteilen.

Nach Ansammlung der Daten, werden diese gefiltert, um einen strukturierten Datensatz zu bilden und Fehlinformationen auszuschließen. Besonders die Stichwortliste lässt zunächst eine große Fehlermage zu, um generelle Erkenntnisse für weitere Ursachen von Fehlinterpretationen zu erkennen und in die Filterung aufzunehmen. Ein detaillierter Ablauf der Filterung ist in Abschnitt 3.2 zu finden. Mit dem gefilterten Datensatz werden abschließend Tabellen und Grafiken zur Analyse der quantitativen Recherche erstellt.

---

[1] Informa TechTarget, o.J., o.S.

[2] Millington, 2019

[3] Valve Corporation, o.J., o.S.

[4] RAWG, o.J., o.S.

Die Hypothese zur quantitativen Analyse leitet sich unmittelbar aus der ersten Teilfrage der Leitfrage ab: „Was ist der Zusammenhang zwischen Algorithmen und Genres?“ Ziel der Überprüfung ist es herauszufinden, ob die Zuordnung von Genres relevanter Spiele zu den genutzten Algorithmen ein Muster erkennen lässt, das darauf hinweist, dass bestimmte Algorithmen genrespezifisch bevorzugt werden.

Da es sich bei beiden untersuchten Variablen um kategorische Daten handelt, wird ein Chi-Quadrat-Unabhängigkeitstest eingesetzt. Der Chi-Quadrat-Unabhängigkeitstest wird häufig genutzt, um in empirischen Untersuchungen Abhängigkeiten zwischen zwei nominalskalierten Variablen sichtbar zu machen. Er dient somit als statistisches Werkzeug, um festzustellen, ob die beobachteten Unterschiede in den Häufigkeiten zufällig entstanden sind oder auf einen systematischen Zusammenhang hindeuten. Dieser Test prüft, ob die beobachteten Häufigkeiten in einer Kontingenztafel (Genre × Algorithmus) signifikant von den erwarteten Häufigkeiten unter Annahme der Unabhängigkeit abweichen. Voraussetzung für die Anwendung ist, dass die erwarteten Zellenhäufigkeiten ausreichend groß sind. Für den Test in dieser Arbeit wurde eine mindeste Zellenhäufigkeit von ≥ 5 festgelegt.
Die zugrundeliegenden Hypothesen lauten somit:

$$H_0: P(Algorithmus \mid Genre) = P(Algorithmus) \quad \forall Genre$$

Die Nullhypothese (H_0) nimmt an, dass kein statistisch signifikanter Zusammenhang zwischen Genre und Algorithmus besteht. Das bedeutet, die Wahrscheinlichkeit der Nutzung eines Algorithmus ist unabhängig vom Genre.

$$H_1: P(Algorithmus \mid Genre) \neq P(Algorithmus) \quad \ni Genre$$

Die Alternativhypothese (H_1) nimmt an, dass ein statistisch signifikanter Zwischen Genre und Algorithmus besteht. Das bedeutet, Es gibt mindestens ein Genre, bei dem die Verteilung der Algorithmen von der erwarteten, unabhängigen Verteilung abweicht.

Da im Rahmen dieser Arbeit untersucht wird, ob ein Zusammenhang zwischen Genre und Algorithmen besteht, liegt der Fokus auf der Alternativhypothese (H_1). Da für den Chi-

Quadrat-Unabhängigkeitstest das Prinzip der Falsifikation gilt, wird zunächst die Nullhypothese (H_0) angenommen. Diese lässt sich einfacher überprüfen. Kann die Nullhypothese (H_0) mit den vorliegenden Daten bestätigt werden, so wird sie verworfen. In diesem Fall ist schlussfolgernd die Alternativhypothese (H_1) anzunehmen.

1.4.2 Qualitativer Teil

Die Experteninterviews auf den GDD bilden das Fundament der qualitativen Analyse. Dieser Teil dient der Beantwortung der zweiten Teilfrage: „Wie werden Algorithmen angewandt?“. Als Ausgangspunkt für die Befragung wurde den Interviewpartnern die Kategorisierung nach Millington präsentiert. Diese Darstellung sollte sicherstellen, dass die Experten auf einem gemeinsamen begrifflichen Fundament aufbauen können und gleichzeitig die Möglichkeit erhalten, bestehende Kategorien kritisch zu erweitern. Es wurde darauf geachtet, keine Ergebnisse aus der quantitativen Recherche den Experten preiszugeben, um den Experten Raum für individuelle Einschätzungen zu geben und nicht durch Vorannahmen zu beeinflussen.

Die Interviewfragen wurden direkt aus den Ergebnissen der quantitativen Analyse abgeleitet und zielten darauf ab, vertiefende Erklärungen zu liefern. Im Fokus standen vier zentrale Fragestellungen: Zum einen wurde untersucht, ob den Experten weitere Kategorien einfallen, die über die Einordnung von Miller hinausgehen. Darüber hinaus wurde erfragt, welche Algorithmen nach Einschätzung der Experten als besonders essentiell gelten, welche dieser Algorithmen tatsächlich in der Praxis Verwendung fanden und in welchen konkreten Kontexten sie eingesetzt wurden. Die Auswahl dieser Fragen erfolgte, um Lücken aus der quantitativen Analyse zu schließen, die zwar Häufigkeit und Muster aufzeigen konnte, jedoch keine kontextuelle Begründung für diese lieferte.

Die qualitativen Interviews verfolgen somit das Ziel, die ermittelten Ergebnisse zu untermauen und ein tieferes Verständnis für die praktische Relevanz der identifizierten Algorithmen zu entwickeln. Durch den Fokus auf konkrete Nutzungsszenarien wird sichergestellt, dass nicht nur abstrakte Häufigkeit, sondern auch praxisorientierte Einschätzungen aus der Industrie in die Analyse mit einfließen. Auf diese Weise lassen sich die quantitativen Befunde interpretieren, ohne die Neutralität der Ausgangsdaten zu beeinträchtigen.

2 Grundlagen

Um zu verstehen, was relevante Algorithmen sind und was damit in dieser Arbeit gemeint ist werden beide Begriffe einzeln betrachtet. Es werden Definitionen der beiden Begriffe erstellt, die im Kontext zum Thema der Arbeit stehen. Da Relevanz und Algorithmen weitreichende Bedeutungen vorweisen, ist diese Eingrenzung beider Begriffe von Nöten.

2.1 Definition von Relevanz

Unter Relevanz von Algorithmen in der Videospielentwicklung wird in dieser Arbeit ein mehrdimensionales Konzept verstanden, das sich nicht allein auf die Literatur oder ihre technische Realisierbarkeit stützt, sondern aus drei miteinander verknüpften Perspektiven bestimmt wird.

1. Algorithmen sind als relevant einzustufen, wenn sie in der Praxis der Spieleentwicklung häufig eingesetzt werden. Die Häufigkeit der Nutzung verweist dabei auf ihre breite Anwendbarkeit in unterschiedlichen Genres, Engines und Spielmechaniken. Ein Algorithmus, der sich in einer Vielzahl von Projekten wiederfindet, hat seine Relevanz dadurch nachgewiesen, dass er ein wiederkehrendes Problem effizient löst und somit über individuelle Produktionen hinaus als Standardwerkzeug etabliert ist. Quantitative Analysen ermöglichen es, solche Trends sichtbar zu machen und die Nutzungshäufigkeit objektiv zu bestimmen.

2. Relevanz wird dadurch definiert, dass ein Algorithmus in erfolgreichen und kulturell prägenden Spielen verwendet wurde. Erfolg wird in diesem Zusammenhang nicht ausschließlich finanziell gemessen, wenngleich kommerzieller Erfolg oftmals ein Indikator für Reichweite und Sichtbarkeit ist. Entscheidend ist vielmehr, ob ein Spiel kulturelle Bedeutung entfaltet und die Gaming-Landschaft nachhaltig beeinflusst. Da die Game Awards Kriterien wie Innovation, Einfluss auf die Spielkultur und Resonanz in der Fachwelt preisen, werden alle nominierten Spiele dieser Auszeichnung als Bezugspunkt genutzt.

3. Schließlich bemisst sich Relevanz and der Komplexität der Implementierung und der damit verbundenen Notwendigkeit, dass Entwickler ein tiefgehendes Verständnis des Algorithmus besitzen. Ein Unterschied besteht hierbei zwischen dem bloßen Einsatz

eines Algorithmus als fertiges Tool durch Anwender, wie Designer und der tatsächlichen Integration auf Code-Ebene durch Programmierer. Nur wenn ein Algorithmus so beschaffen ist, dass er in der spezifischen Architektur einer Game Engine bewusst angepasst, optimiert oder erweitert werden kann, zeigt sich seine Relevanz für die Videospielentwicklung im engeren Sinn. Diese Dimension hebt Relevanz von rein oberflächlich eingesetzten Verfahren ab und macht deutlich, welche Algorithmen für die Ausbildung und Praxis zukünftiger Entwickler unverzichtbar sind.

Zusammenfassend wird Relevanz in dieser Arbeit also nicht eindimensional verstanden, sondern als Zusammenspiel von Nutzungshäufigkeit, kultureller und branchenspezifischer Bedeutung sowie technischer Tiefe der Implementierung. Erst das Zusammenspiel dieser drei Faktoren erlaubt eine fundierte Bewertung, welche Algorithmen im Kontext moderner Videospielentwicklung als zentral angesehen werden.

2.2 Definition von Algorithmus

Für die vorliegende Arbeit wird der Begriff der Algorithmus im Kontext der Videospielentwicklung spezifisch auf den Bereich der Spielmechanik bezogen. Unter einem Algorithmus wird im Allgemeinen ein schrittweises Verfahren verstanden, das eine Lösung zu einem gegebenen Problem hervorbringt. Millington definiert dies pregnant: „Algorithms are step-by-step processes that generate a solution to an AI problem “.[1] Diese Definition bildet den Ausgangspunkt der Arbeit, da sie die funktionale Natur von Algorithmen betont und zugleich den Bezug zur Künstlichen Intelligenz (KI) in Spielen herstellt.

Im Rahmen der Spieleentwicklung bezeichnet der Begriff AI oder KI nicht ausschließlich das moderne Verständnis von Künstlicher Intelligenz, wie es vor allem durch generative Sprachmodelle oder Machine-Learning-Systeme geprägt ist. Vielmehr umfass AI in der Videospielentwicklung sämtliche Verfahren, die das Verhalten von Systemen und Nicht-Spieler-Charakteren (NPCs) steuern, ohne dass diese unmittelbar durch den Input der Spielenden ausgelöst werden. Hierzu zählen sämtliche Mechanismen, die notwendig sind, um die Spielwelt glaubwürdig, dynamisch und unterhaltend zu gestalten. Damit unterschiedet sich die Spiel-KI von der in der Informatik oder im Machine Learning

[1] (Millington, 2019, S. 14, Abschnitt 1.3.1)

gebräuchlichen Auffassung, die primär auf das selbstständige Lernen und Generieren von Inhalten abzielt. Zur Systematisierung der in Videospielen eingesetzten Algorithmen bietet die von Millington etablierte Klassifikationen einen geeigneten Rahmen. Er unterscheidet die folgenden Überkategorien oder auch Arten von AI: Movement, Pathfinding, Decision Making, Tactical and Strategic AI, Learning AI, Procedual Content Generation und Board Games. Diese Kategorien machen deutlich, dass Algorithmen für Spiele-KI eine weite Spannbreite an Funktionen bedecken, die von der Mikroebene der Bewegungssteuerung bis zur Makroebene langfristiger Strategien reichen.

Für diese Arbeit ausdrücklich ausgeschlossen sind hingegen Algorithmen, die speziell für die Erstellung von Grafiken genutzt werden. Zwar stellen auch diese einen wesentlichen Teil der Videospielentwicklung dar, doch konzentrieren sie sich auf die visuelle Darstellung und Optimierung von Bildern, Animationen und Effekten. Der hier zugrunde gelegte Algorithmus Begriff beschränkt sich hingegen auf spielmechanische KI, also Verfahren, die das Verhalten des Spiels und seiner nicht unmittelbar durch Spielereingaben kontrollierten Elemente bestimmen.

Zusammenfassend wird ein Algorithmus im Rahmen dieser Arbeit als formalisiertes, schrittweises Verfahren zur Lösung eines Problems im Bereich der spielmechanischen Künstlichen Intelligenz, das insbesondere das Verhalten von NPCs, die Dynamik der Spielwelt und die Generierung spielbezogener Inhalte steuert.

3 Quantitative Recherche

Um die Relevanz nachzuweisen, wurden alle jemals nominierten Spiele der Game Awards als Datensatz ausgewählt. Die Auszeichnungen der Game Awards werden von einer internationalen Jury aus Medienvertretern vergeben und gelten als eine der prestigeträchtigsten Anerkennungen der Branche.[1] Die Game Awards existieren seit 2014 und finden im Dezember statt. Das heißt die Zeitspanne aller nominierten Spiele der Game Awards umfasst die letzten 10 Jahre von 2014 bis 2024. Dieser Zeitraum wurde gewählt, da AAA-Spiele in ihrer Entwicklung meist drei bis fünf Jahre brauchen.[2] Bei einer maximalen Entwicklungszeit von fünf Jahren, kann so bei zwei Veröffentlichungen einer AAA-Spielereihe ein Trend beobachtet werden.

3.1 Vorbereitung der Daten

Die Liste der nominierten Spiele wurde Artikeln entnommen.[3] Nominierungen in Kategorien, die keine Spiele enthalten, beispielsweise „Best Performace“[4] wurden ausgeschlossen. Auf diese Weise gehen 648 Eindeutige Einträge hervor. Um nun herauszufinden welche Videospiele welche Algorithmen nutzen und welchem Genre zuzuordnen sind wurden zwei Webseiten automatisiert ausgewertet.

Um zu bestimmen, welche Algorithmen in den nominierten Spielen verwendet werden, wurde die Liste der Spiele zunächst auf eindeutige Einträge reduziert, da einige Titel in mehreren Jahren nominiert waren. Dies ergibt 648 eindeutige Spiele. Als Quelle für die Analyse wurde die Online-Fachzeitschrift Game Developer[5] gewählt. Die Plattform bietet eine umfangreiche Sammlung an Artikeln von Entwicklerinnen und Entwicklern sowie Redakteuren, die speziell auf die Bedürfnisse von Spieleentwicklern ausgerichtet sind. Zudem erlaubt die gleichmäßige Struktur der Website ein systematisches Erfassen von Informationen. Die Suchfunktion der Game Developer-Seite enthält einen „Programming“-Filter, der die Suche gezielt auf Artikel im Bereich der Programmierung einschränkt.

---

[1] vgl. Mcwhertor, 2024, o.S; The Game Awards, 2025, o.S.

[2] vgl. INNOVECS Games, 2025, o.S.

[3] Farell, 2024, o.S.; Turtle Beach, 2023, o.S.; Harte, 2022, o.S.; Stewart, 2021, o.S.; Park, 2020, o.S.; Plant, 2019, o.S.; Reeves, 2018, o.S.; Makuch, 2017, o.S.; Makuch, 2016, o.S.; Sarkar, 2015, o.S.; Makuch, 2014, o.S.

[4] Farell, 2024, o.S.

[5] Informa TechTarget, o.J., o.S

Zunächst wurden alle Artikel gesammelt, in denen ein Spieletitel aus der eindeutigen Liste der nominierten Spiele auftauchte und die im „Programming“-Filter erscheinen. Auf diese Weise entstand ein Datensatz von 2.475 Artikeln. Von den 648 Spielen konnten auf diese Weise Artikel zu 315 Spielen gefunden werden. Weitere Recherchen auf alternativen Websites waren nicht erfolgreich, da viele Quellen keine einheitliche Struktur aufweisen oder stark werbelastig bzw. spekulativ sind. Aus diesem Grund wurde arbeitet die weitere Analyse mit dem Datensatz der 315 Spiele, für die Artikel gefunden wurden.

Für die Extraktion relevanter Algorithmen wurde zunächst eine Stichwortliste erstellt, die auf dem Inhaltsverzeichnis von AI for Games basiert. Die Begriffe wurden in der Singularform erfasst und in einigen Fällen in ihre Bestandteile zerlegt, um eine präzisere Suche nach Fachtermini zu ermöglichen. Die resultierende Rohfassung umfasst 401 Stichwörter. Diese wurden in sieben „AI-Kategorien“ gegliedert, die unmittelbar aus den Kapiteln von AI for Games nach Millington übernommen wurden.[1] Millington unterscheidet dabei die Kategorien Movement, Pathfinding, Decision Making, Tactical and Strategic AI, Learning, Procedural Content Generation sowie Board Games. Im weiteren Text wird mit dem Begriff „AI-Kategorien“ stets auf diese Einteilung verwiesen. Die unverarbeitete Stichwortliste wurde anschließend genutzt, um alle 2.475 Artikel nach den definierten Begriffen zu durchsuchen. Dieser Zwischenschritt erfolgte bewusst vor einer inhaltlichen Filterung, um potenzielle Missverständnisse oder Probleme im automatisierten Erkennungsprozess identifizieren zu können.

Für die Zuordnung der Spiele zu Genres wurde die RAWG-Datenbank genutzt. Im Gegensatz zu Steam, deren Genre-Zuordnungen nur PC-Spiele abdecken, enthält RAWG alle Spiele plattformübergreifend und ermöglicht somit eine vollständige Genrezuordnung der 648 nominierten Spiele. RAWG beinhaltet über 500.000 Spiele auf mehr als 50 Plattformen und gilt damit als größte plattformübergreifende Datenbank. Zwar werden die genauen Quellen der Daten nicht offengelegt und RAWG hat auf diesbezügliche Anfragen bisher nicht geantwortet, jedoch weist die API-Dokumentation auf eine Kombination aus Nutzerbeiträgen und algorithmischen Verfahren hin.[2] Die Plattform wird zudem von zahlreichen vertrauenswürdigen Unternehmen genutzt. RAWG erlaubt bis zu 20.000

[1] (Millington, 2019, S. viii-xvi,S. 13, Abschnitt 1.2.6)

[2] RAWG,RAWG API Dokumentation o.J.

Application Programming Interface (API)-Aufrufe pro Monat, was für die Analyse der 648 Spiele ausreicht, und die kostenlose Version gestattet die Nutzung für nicht-kommerzielle Projekte.[1] Für alle Spiele konnte mindestens ein Genre ermittelt werden. Es wurde eine Spiel-Genre-Liste erstellt, die zunächst keiner weiteren Verarbeitung bedarf.

3.2 Aufbereitung der Daten

Für die Analyse war es notwendig, die erstellte Stichwortliste systematisch aufzubereiten. Während die ungefilterte Liste von 401 Einträgen zunächst eine breite Grundlage für die Suche in den Artikeln darstellte, zeigten sich bei der Auswertung verschiedene Probleme. Daher wurden im weiteren Verlauf mehrere Schritte unternommen, um die Datengrundlage zu bereinigen und die Zuverlässigkeit der Ergebnisse zu erhöhen. Die folgenden Abschnitte beschreiben zunächst die Schwierigkeiten der ursprünglichen Liste, bevor anschließend die methodische Filterung erläutert wird.

Die unverarbeitete Stichwortliste erwies sich aus mehreren Gründen als problematisch. Nicht alle Begriffe lassen sich eindeutig einer Kategorie zuordnen, zudem traten zahlreiche Dopplungen auf. Hinzu kommt, dass einige der Begriffe mehrdeutig sind und auch im alltäglichen englischen Sprachgebrauch vorkommen, was die Gefahr von Fehlzuordnungen erhöht. Darüber hinaus enthalten manche Stichwörter andere Begriffe, was in Einzelfällen zu einer Verfälschung der ursprünglichen Bedeutung führen kann. So wird beispielsweise das Stichwort „Face“ innerhalb des Wortes „surfaced“ fälschlicherweise erkannt. Allerdings gilt dies nicht uneingeschränkt, da es auch Gegenbeispiele gibt: Im Fall von „Movement“ in „Movementsystem“ bleibt die ursprüngliche Bedeutung erhalten, sodass weiterhin eine korrekte Zuordnung zu einer Kategorie möglich ist. Diese Beispiele verdeutlichen, dass eine differenzierte Filterung erforderlich ist, um die Qualität der Datengrundlage sicherzustellen.

Um die beschriebenen Probleme zu reduzieren, wurde eine systematische Filterung der Stichwortliste vorgenommen. Zunächst wurden mithilfe des im Anhang D dokumentierten Skripts doppelte Einträge identifiziert, die in mehreren Kategorien vorkamen, und aus dem Datensatz entfernt. Anschließend erfolgte eine Aufteilung in zwei Teildatensätze: Der erste enthielt ausschließlich Stichwörter, die innerhalb anderer Begriffe vorkamen, während der zweite nur eindeutige Einträge umfasste. Von dieser Regel ausgenommen wurden Fälle, in

[1] RAWG, o.J., o.S.

denen lediglich das Plural-s angehängt war, da dies im Englischen die Bedeutung nicht verfälscht. Das für diese Trennung verwendete Skript ist im Anhang E dargestellt. Um zudem sicherzustellen, dass nur Fachbegriffe berücksichtigt werden und keine Verfälschungen durch den allgemeinen Sprachgebrauch entstehen, wurde eine zusätzliche Filterung anhand einer Comma-Seperated-Values (CSV)-Datei der Oxford 5000 vorgenommen.[1] Diese Wortliste stellt eine erweiterte Kernwortsammlung für fortgeschrittene Englischlernende dar und enthält neben den Oxford 3000 weitere 2.000 hochfrequente Wörter, die sich am Gemeinsamen Europäischen Referenzrahmen (CEFR)[2] orientieren und typischerweise von Sprecherinnen und Sprechern auf dem Niveau B2–C1 genutzt werden.[3] Da die Begriffe in neutraler Form dokumentiert sind, wurden Endungen wie „-ing“, „-ies“, „-es“, „-ed“ und „-s“ vor der Filterung entfernt. Dieser Schritt wurde mithilfe des im Anhang F dargestellten Skripts automatisiert. Als letzter Ansatz wurde zudem versucht, nur Artikel zuzulassen, in deren Titel ein Spieletitel vorkam. Diese Einschränkung reduzierte den Datensatz jedoch von 2.475 Artikeln auf lediglich 97 Artikel zu 47 Spielen, was für die Analyse eine zu geringe Basis darstellte. Nach Abschluss der beschriebenen Filterung ergab sich ein Datensatz von 215 Spielen, denen jeweils mindestens ein Algorithmus aus einer der von Millington definierten Kategorien zugeordnet werden konnte.

Trotz der systematischen Filterung konnten nicht alle Probleme der ursprünglichen Stichwortliste behoben werden, und es traten zudem neue Schwierigkeiten auf. So gingen beispielsweise Stichwörter verloren, deren ursprüngliche Bedeutung nicht verfälscht war, obwohl sie Teil eines anderen Wortes sind. Dieses Problem konnte durch die automatisierte Filterung nicht gelöst werden. Darüber hinaus werden Begriffe, die zwar im alltäglichen englischen Sprachgebrauch vorkommen und somit in der Oxford 5000 enthalten sind, aber dennoch eindeutig einer AI-Kategorie zugeordnet werden könnten, wie etwa „Movement“, nicht berücksichtigt. Diese Einschränkung stellt ein neues Problem dar. Angesichts der Vielfalt und Komplexität der identifizierten Schwierigkeiten lässt sich erkennen, dass eine ausschließlich systematische Filterung nicht ausreicht, weshalb die Erstellung einer kuratierten Stichwortliste als notwendig erachtet wird.

---

[1] Zhiyanov, 2021

[2] vgl. Oxford University Press, *Ocford Learner's Dictionaries / What is the CERF*,o.J, o.S.

[3] vgl. Oxford University Press, o.J., o.S.

Um die verbleibenden Probleme der systematischen Filterung zu beheben, wurde eine kuratierte Stichwortliste erstellt. Bei der Zusammenstellung wurden gezielt Regeln angewendet, um die Eindeutigkeit und Aussagekraft der Einträge sicherzustellen. Nur Begriffe, die eine klare Zuordnung zu einer AI-Kategorie erlauben, werden aufgenommen, um Mehrdeutigkeiten zu vermeiden. Dopplungen werden ausgeschlossen, um Redundanzen zu verhindern, und alle Einträge verbleiben in der Singularform, da Pluralformen keine zusätzliche Information liefern und die Analyse erschweren könnten. Begriffe, die Teil eines anderen Wortes sind, deren ursprüngliche Bedeutung jedoch unverfälscht bleibt, werden wieder aufgenommen, um relevante Einträge nicht zu verlieren. Ebenso werden bestimmte Stichwörter aus der Oxford 5000 berücksichtigt, wenn sie trotz ihres allgemeinen Sprachgebrauchs eindeutig einer Algorithmuskategorie zugeordnet werden können. Durch die Anwendung dieser Regeln wird gewährleistet, dass die kuratierte Liste präzise, konsistent und für die Zuordnung der Spiele zu den von Millington definierten AI-Kategorien geeignet ist. Die resultierende kuratierte Einteilung der Stichwörter, die tatsächlich auch gefunden wurden, wird in der entsprechenden Tabelle dargestellt.

Kategorie	Stichwort
Movement	"Movement", "Steering", "Collision Avoidance", “Obstacle Avoidance”, "Jumping", “Shooting”, "Coordinated Movement", "Motor Control", "(enemy)Movement()", "(vehicle)Movement(stdboat)", "(unit)Movement()"
Pathfinding	"Pathfinding", "Dijkstra", "A*", "Navigation Mesh", "Hierarchical Pathfinding", "Navmesh", "(spatialquery)Navmesh(sample)", "(pimplrecast)Navmesh()"
Decision Making	"Decision Making", "Decision Tree", "Finite State Machine", "State Machine", ”Hierarchical State Machine”, "Behavior Tree", "Fuzzy Logic"," Goal-oriented Action Planning", "Tactical and Strategic AI"
Tactical and Strategic AI	"Tactical AI", "Strategic AI", "Influence Map", "Convolution Filters", "Cellular Automata"
Learning	"Annealing", "String Matching", "ID3", "Reinforcement Learning", "Deep Learning", "generative AI", “Artificial Neural Network”
Procedual Content Generation	"Procedural Content Generation", "Poisson Disk", "Landscape Generation",< "Height-Map", "Maze Generation",
Board Games	"Game Theory", "Board Game",

Tabelle 1: Gefundenen Stichwörter der kuratierten Liste, eingeteilt in die AI-Kategorien nach Millington

Nachdem überflüssige Einträge mit dem Skript aus Anhang F entfernt wurden, entsteht ein Datensatz aus 207 Spielen, denen mindestens ein Algorithmus aus der Tabelle *1* zugeordnet werden konnte.

3.3 Beobachtung der Genres

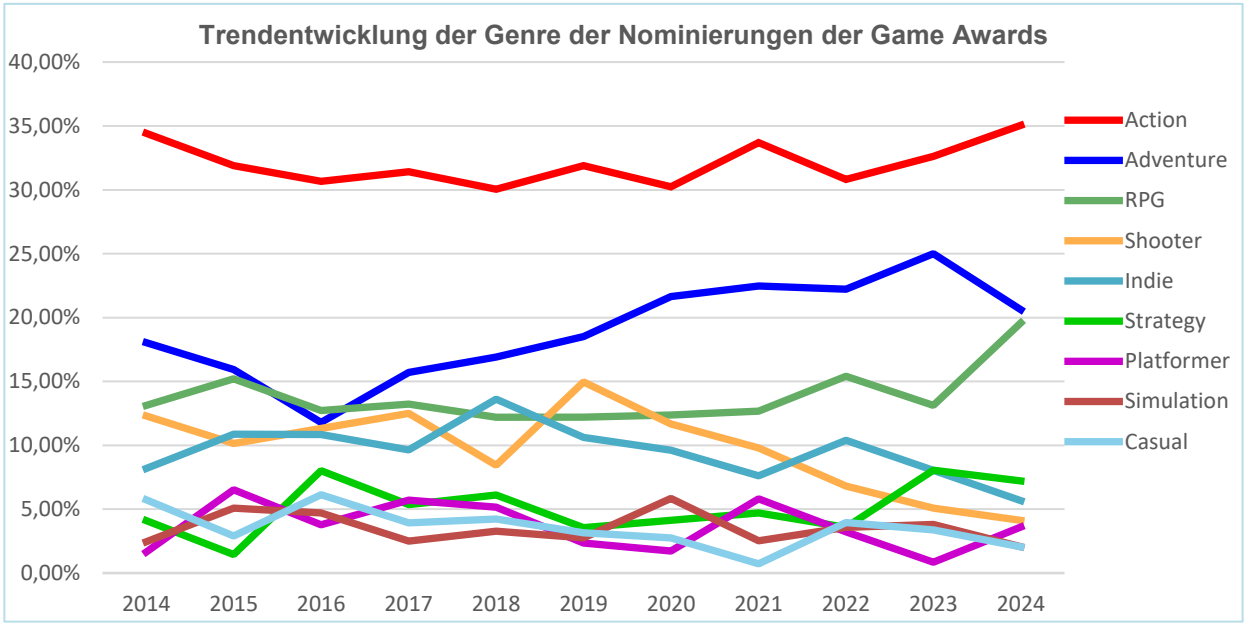

Abbildung 2: Trendentwicklung der Genres aller nominierten Spiele bei den Game Awards 2014–2024 (eigene Darstellung)

Die Auswertung der Game Awards-Nominierungen von 2014 bis 2024, wie sie in Abbildung 2 zu sehen ist, zeigt deutliche Unterschiede in der Entwicklung der einzelnen Genres. Am konstantesten und zugleich dominierend bleibt das Action-Genre. Mit Anteilen zwischen rund 30 % und 35 % stellt es über den gesamten Zeitraum hinweg die größte Kategorie und konnte in den letzten Jahren seine Position weiter ausbauen.

Adventure-Spiele haben seit 2016 eine kontinuierlich wachsende Präsenz im Nominierungsfeld gezeigt. Während ihr Anteil 2016 noch bei rund 11,79 % lag, stieg er in den darauffolgenden Jahren stetig an. Erst 2024 kam es erstmals seit 2016 wieder zu einem Rückgang, wobei der Anteil von 2023 noch bei 25% auf 20,62 % fiel. Trotz dieser leichten Abnahme bleiben Adventure-Spiele weiterhin stark vertreten und belegen knapp den zweiten Platz unter den Genres.

Rollenspiele (RPGs) zeigen eine bemerkenswerte Dynamik: Nach einem gleichbleibenden Anteil bis 2019 ist ab 2020 ein deutlicher Aufwärtstrend zu beobachten, der 2023 und 2024 seinen Höhepunkt erreicht. Damit etablieren sich RPGs zunehmend als dritte Leitkategorie neben Action und Adventure.

Shooter verzeichnen dagegen einen klaren Abwärtstrend. Nachdem sie 2019 mit fast 15 % ihren Höchstwert erreichten, sinkt ihr Anteil seitdem kontinuierlich und liegt 2024 nur noch bei knapp 4 %. Dieses Genre verliert damit spürbar an Bedeutung im Kontext der Nominierungen.

Strategy-Spiele entwickeln sich gegenläufig: Sie lagen lange Zeit zwischen 1 % und 6 %, erreichten jedoch 2023 mit 8,05 % ihren Höchstwert und blieben 2024 mit 7,22 % weiterhin auf einem vergleichsweise hohen Niveau. Dies deutet auf eine wachsende Relevanz des Genres hin.

Indie-Spiele hingegen verlieren an Bedeutung. Nachdem sie in den frühen Jahren noch zweistellige Werte erreichten, liegen sie ab 2020 fast durchgehend im Bereich von 5–7 %. Ihr relativer Einfluss nimmt damit ab.

Die restlichen Genres – darunter Casual, Simulation, Fighting, Plattformspiele, Sport, Arcade, Racing, Puzzle sowie kleinere Kategorien wie Family-, Card- oder Board-Games – bewegen sich überwiegend im Bereich von 5 % oder darunter. Sie treten damit zwar regelmäßig in Erscheinung, ohne jedoch die Relevanz der dominierenden Genres zu erreichen.

Es ist jedoch zu beachten, dass die Kategorien der Game Awards nicht über den gesamten Zeitraum hinweg unverändert bleiben. Jährlich können neue Kategorien hinzugefügt oder bestehende entfernt werden. Die dargestellten Entwicklungen spiegeln somit in erster Linie die Trends innerhalb des Nominierungsrahmens der Game Awards wider und dürfen nicht unmittelbar, als Abbild des gesamten Spielemarktes interpretiert werden.

Zusammenfassend lässt sich festhalten, dass Action- und Adventure-Spiele im Untersuchungszeitraum ihre Vormachtstellung behaupten konnten, während Rollenspiele und Strategy-Games an Sichtbarkeit gewannen. Shooter und Indie-Spiele verlieren dagegen deutlich an Bedeutung.

3.4 Beobachtung der Algorithmen

Zur Beobachtung der Algorithmen wurde auf Basis der kuratierten Stichwortliste[1] überprüft, ob in den zugeordneten Artikeln zu einem Spiel mindestens ein Begriff einer der definierten Kategorien vorkam. Sobald ein Stichwort erkannt wurde, galt das Spiel als Träger dieser Kategorie, das heißt, die jeweilige AI-Kategorie wurde diesem Spiel zugewiesen. Auf diese Weise konnte eine Häufigkeitsverteilung erstellt werden, die angibt, wie viele Spiele in der Stichprobe pro Kategorie mindestens einen Algorithmus enthalten.

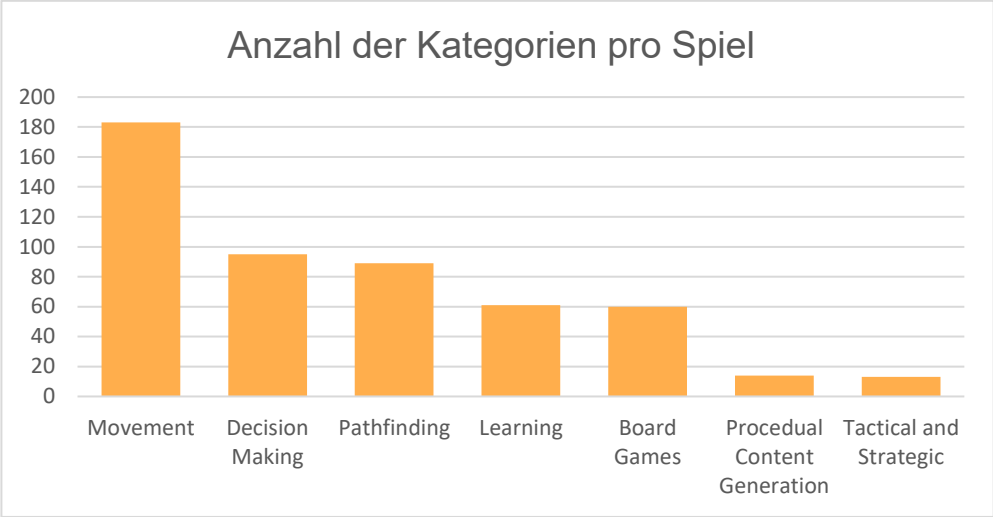

Abbildung 3: Anzahl Kategorien pro Spiel, welche mindestens ein Algorithmus vorweisen (eigene Darstellung)

Das Ergebnis ist in Abbildung *3* dargestellt. Die Gesamtheit der Zuweisungen umfasst 515 Einträge. Am häufigsten tritt die Kategorie Movement auf, die 183 Nennungen erreicht und damit 37,10 % aller identifizierten Vorkommen ausmacht. An zweiter Stelle folgt Decision Making mit 95 Nennungen (21,89 %), dicht gefolgt von Pathfinding mit 89 Nennungen (20,51 %). Die Kategorie Learning liegt mit 61 Nennungen bei 14,06 % und bildet damit den viertgrößten Anteil. Deutlich geringer fallen die Kategorien Procedural Content Generation (3,23 %) und Tactical and Strategic AI (3,00 %) aus. Mit 60 Nennungen (11,65 %) weist Board Games einen nahezu gleich großen Anteil auf. Deutlich geringer vertreten sind Procedural Content Generation (2,72 %) und Tactical and Strategic AI (2,52 %). Die Darstellung macht deutlich, dass einzelne Kategorien weitaus häufiger identifiziert wurden als andere.

[1] Tabelle *1*: Gefundenen Stichwörter der kuratierten Liste, eingeteilt in die AI-Kategorien nach Millington

Gleichwohl ist zu berücksichtigen, dass die Ergebnisse unmittelbar an die Methodik der Datenerhebung gebunden sind. Die Erfassung basiert ausschließlich darauf, ob ein vordefiniertes Stichwort in einem Artikel zu einem bestimmten Spiel auftauchte. Somit besteht die Möglichkeit, dass Algorithmen in Spielen zwar verwendet, jedoch nicht explizit benannt wurden und dadurch in dieser Zählung nicht erscheinen. Ebenso kann es vorkommen, dass Stichwörter mehrfach oder in anderem Kontext genannt wurden, wodurch eine Kategorie einem Spiel zugewiesen wurde, ohne dass dies zwangsläufig die tatsächliche Implementierung widerspiegelt. Diese Einschränkungen sind bei der Interpretation der Daten zu beachten.

Neben der Gesamthäufigkeit der Kategorien wurde zusätzlich die jährliche Entwicklung betrachtet. Grundlage hierfür ist dieselbe Vorgehensweise wie zuvor: Ein Spiel wurde einer Kategorie zugeordnet, sobald in den Artikeln mindestens ein Stichwort der entsprechenden Kategorie identifiziert werden konnte. Im Unterschied zur Gesamtauswertung werden die Ergebnisse hier jedoch nach Jahr aufgeschlüsselt. Dadurch ergibt sich ein prozentualer Überblick darüber, wie viele der in einem Jahr untersuchten Spiele mit einer bestimmten Kategorie verknüpft wurden.

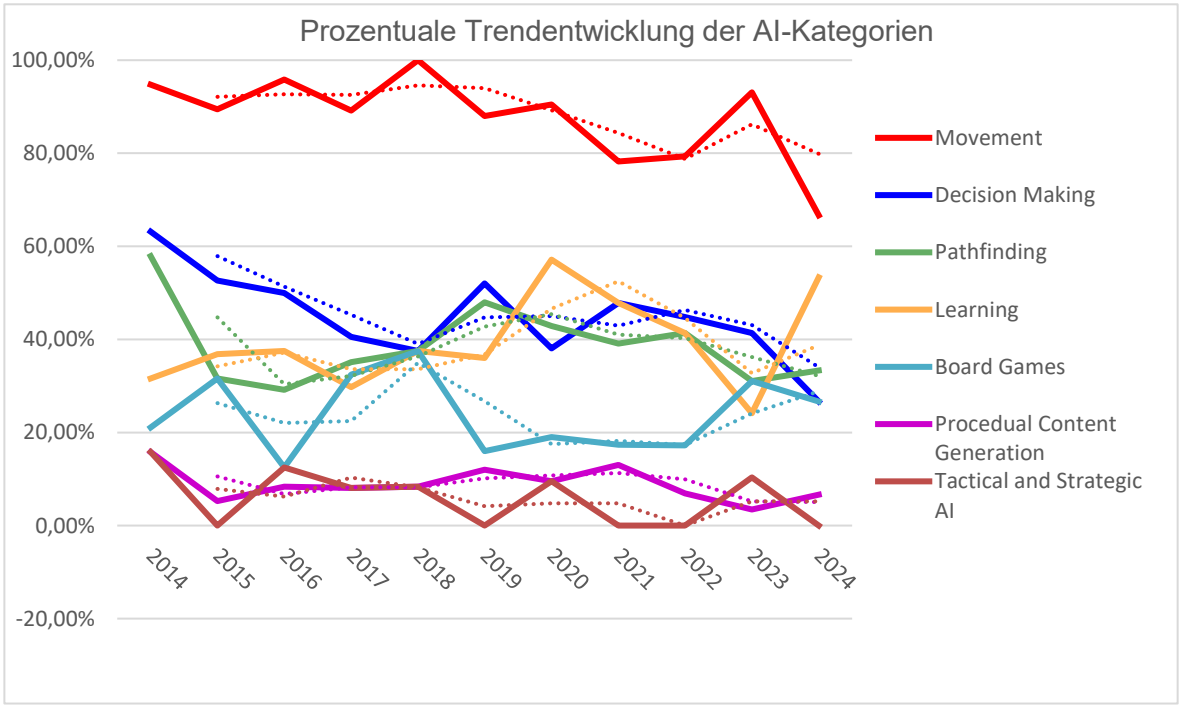

Abbildung 4: Prozentuale Trendentwicklung der AI-Kategorien in den Jahren 2014-2024 (eigene Darstellung)

Die in Abbildung *4* dargestellten Ergebnisse verdeutlichen, dass die Verteilung der Kategorien über die Jahre hinweg deutliche Unterschiede aufweist. Die Kategorie Movement ist in nahezu allen Jahren mit sehr hohen Anteilen vertreten und überschreitet überwiegend 85 %, mit einzelnen Ausnahmen (z. B. 2021 und 2024). Decision Making und Pathfinding zeigen im Zeitverlauf eine mittlere Ausprägung und bewegen sich in der Regel zwischen etwa 30 % und 60 %. Die Kategorie Learning nimmt bis 2019 überwiegend Werte im Bereich von rund 30 % bis 40 % ein, zeigt jedoch ab 2020 deutlich höhere Werte und erreicht in mehreren Jahren Anteile von über 50 %.

Die Kategorie Board Games tritt über die Jahre hinweg zwar regelmäßig auf, bleibt jedoch durchgehend im unteren bis mittleren Bereich (zwischen 12 % und 37 %). Procedural Content Generation sowie Tactical and Strategic AI sind dagegen nur mit geringen Anteilen vertreten und überschreiten selten 15 %. Insgesamt bestätigt sich damit die Dominanz von Movement über alle Jahre hinweg, während die übrigen Kategorien je nach Jahr stark variieren und teilweise nur in begrenztem Umfang auftreten.

Es ist zu beachten, dass die zugrundeliegende Datenbasis deutlich kleiner ist als beim Genre-Datensatz. Während für die Genre-Analyse 648 eindeutige Spiele einbezogen werden konnten, umfasst der hier betrachtete Datensatz lediglich 207 Spiele, denen mindestens eine AI-Kategorie zugewiesen wurde. Die prozentuale Darstellung in Abbildung *4* bezieht sich somit nur auf diese eingeschränkte Menge und darf nicht mit der Gesamtzahl der nominierten Spiele gleichgesetzt werden. Die Ergebnisse geben daher ausschließlich Auskunft über die relative Präsenz der Kategorien im Zeitverlauf innerhalb dieses kleineren Datensatzes. Eine tabellarische Übersicht ist in Anhang L zu finden.

3.5 Zusammenhang zwischen Genre und Algorithmen

Im vorliegenden Abschnitt wird untersucht, ob ein statistisch signifikanter Zusammenhang zwischen den Spielgenres und den eingesetzten AI-Kategorien besteht. Ziel ist es herauszufinden, ob bestimmte Algorithmen genretypisch häufiger auftreten oder ob ihre Verwendung unabhängig vom Genre erfolgt.

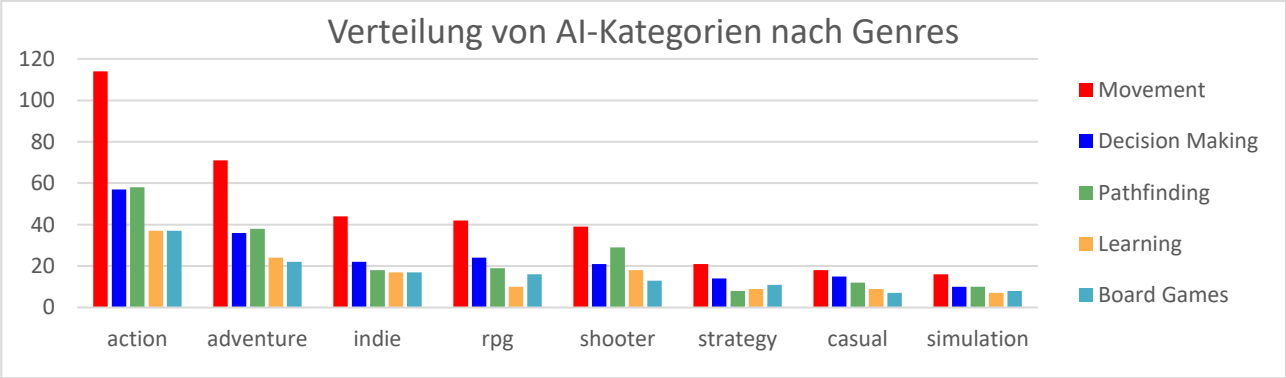

Abbildung 5: Kontingenztafel zwischen Genre und AI-Kategorie (eigene Darstellung)

Anzahl verschiedener Spiele	AI-Kategorie					
Genre	Movement	Decision Making	Pathfinding	Learning	Board Games	Gesamtergebnis
action	114	57	58	37	37	128
adventure	71	36	38	24	22	81
indie	44	22	18	17	17	51
rpg	42	24	19	10	16	50
shooter	39	21	29	18	13	43
strategy	21	14	8	9	11	26
casual	18	15	12	9	7	20
simulation	16	10	10	7	8	20
Gesamtergebnis	166	90	86	60	54	188

Tabelle 2: Kontingenztafel zwischen Genre und AI-Kategorie mit der Anzahl verschiedener Spiele als Wert (eigene Tabelle)

Die Grundlage der Analyse bildet eine Kontingenztafel, in der die Häufigkeiten der einzelnen AI-Kategorien nach Genres aufgeschlüsselt sind. Die entsprechenden Werte wurden in dargestellt. Abbildung *5* zeigt ergänzend die grafische Verteilung der AI-Kategorien innerhalb der Genres. Bemerkenswert ist, dass in den Genres Action, Adventure, Indie und RPG die ersten vier AI-Kategorien in etwa gleichmäßig vertreten sind. Dieses Muster deutet bereits darauf hin, dass sich kein ausgeprägter Zusammenhang zwischen Genre und Algorithmus abzeichnet, wie er im Falle einer klaren Abhängigkeit zu erwarten wäre.

Zur Überprüfung dieser Vermutung wurde der Chi-Quadrat-Unabhängigkeitstest durchgeführt. Da dieser Test voraussetzt, dass in allen Zellen der Kontingenztafel eine erwartete Häufigkeit von mindestens fünf gegeben ist, mussten die Kategorien Tactical and Strategic AI sowie Procedural Content Generation ausgeschlossen werden. Darüber hinaus wurden weitere selten vertretene Genres entfernt, bis diese Bedingung erfüllt war. Auf Basis der reduzierten Daten wurde anschließend eine Erwartungstabelle berechnet, die die Häufigkeitsverteilung unter der Annahme völliger Unabhängigkeit zwischen den Variablen Genre und AI-Kategorien abbildet.

Die Durchführung des Tests erfolgte, indem die beobachteten Häufigkeiten mit den erwarteten Häufigkeiten verglichen und die Abweichungen gemäß der Chi-Quadrat-Formel quadriert und gewichtet wurden. Bei einem Signifikanzniveau von 5 % ergab sich ein Chi-Quadrat-Wert von 0,787. Dieser Wert liegt deutlich unterhalb des kritischen Wertes für die entsprechenden Freiheitsgrade. Damit konnte die Nullhypothese, die von der Unabhängigkeit zwischen Genre und AI-Kategorien ausgeht, nicht verworfen werden.

Das Ergebnis bedeutet, dass in den untersuchten Daten kein statistisch signifikanter Zusammenhang zwischen dem Genre eines Spiels und der eingesetzten AI-Kategorien festgestellt werden konnte. Auch wenn einzelne Genres – wie Action im Bereich Movement – visuell durch stärkere Ausprägungen auffallen, handelt es sich hierbei um zufällige Schwankungen, die nicht auf eine systematische Abhängigkeit schließen lassen. Folglich ist die Verteilung der Algorithmen in den untersuchten Genres als unabhängig zu betrachten. Ergänzend wurde dieselbe Analyse auch auf Ebene der einzelnen Stichwörter durchgeführt, um mögliche Verzerrungen durch die Zusammenfassung in Oberkategorien auszuschließen. Der dabei berechnete Chi-Quadrat-Wert lag bei 0,994, was noch deutlicher zeigt, dass keinerlei statistisch signifikanter Zusammenhang besteht. Dieses Ergebnis bestätigt, dass die Verteilung der Algorithmen unabhängig vom Genre erfolgt und die beobachteten Unterschiede rein zufällig sind.

3.6 Ergebnis der Quantitativen Recherche

Die quantitative Analyse hat gezeigt, dass die Kategorie Movement mit deutlichem Abstand am häufigsten in den untersuchten Spielen vorkommt, gefolgt von Decision Making und Pathfinding. Die Kategorien Learning und Board Games sind zwar ebenfalls vertreten, jedoch in geringerer Ausprägung. Die zunächst berücksichtigten Kategorien Tactical and Strategic AI sowie Procedural Content Generation mussten aufgrund zu geringer Zellenhäufigkeiten von der weiteren Analyse ausgeschlossen werden.

Der durchgeführte Chi-Quadrat-Unabhängigkeitstest ergab keinen statistisch signifikanten Zusammenhang zwischen den untersuchten Genres und den AI-Kategorien. Dies gilt sowohl für die Auswertung auf Ebene der Oberkategorien ($\chi^2 = 0{,}79$; $p > 0{,}05$) als auch für die ergänzende Analyse auf Ebene der einzelnen Stichwörter ($\chi^2 = 0{,}99$; $p > 0{,}05$). Somit ist davon auszugehen, dass die Verteilung der Algorithmen unabhängig vom Genre erfolgt.

Insgesamt lässt sich festhalten, dass bestimmte Algorithmen zwar häufiger in einzelnen Genres auftreten, diese Unterschiede jedoch nicht über zufällige Schwankungen hinausgehen. Ein systematischer Zusammenhang zwischen Genre und AI-Kategorien konnte in der quantitativen Untersuchung nicht nachgewiesen werden.

Aufbauend auf diesen Ergebnissen wird im nächsten Schritt die qualitative Analyse in Form von Experteninterviews durchgeführt. Während die quantitative Recherche lediglich Häufigkeiten und Verteilungen sichtbar machen konnte, zielen die Interviews darauf ab, ein tieferes Verständnis für die Hintergründe dieser Ergebnisse zu gewinnen. Die Fragen sind so konzipiert, dass sie die in der quantitativen Analyse ermittelte Unabhängigkeit zwischen Genre und Algorithmen kritisch hinterfragen und mögliche Erklärungen für die beobachteten Muster liefern. Zugleich sollen die Experteneinschätzungen dazu beitragen, die Ergebnisse zu untermauern oder alternative Interpretationen aufzuzeigen. Auf diese Weise bildet die qualitative Erhebung eine notwendige Ergänzung, um über die rein statistische Betrachtung hinaus auch kontextuelle und praxisnahe Einblicke in die Nutzung von Algorithmen in der Spieleentwicklung zu erhalten.

4 Qualitative Experteninterviews

Die qualitativen Experteninterviews wurden mit dem Ziel geführt, die Ergebnisse der quantitativen Analyse zu vertiefen und um praktische Einblicke aus der Industrie zu ergänzen. Während die quantitative Untersuchung Häufigkeiten und Verteilungen bestimmter Algorithmen aufzeigen konnte, blieb unbeantwortet, wie diese Verfahren konkret in der Praxis ausgewählt und eingesetzt werden. Die Interviews verfolgen somit das Ziel, die zweite Teilfrage der Arbeit, wie Algorithmen angewandt werden, zu adressieren und ein besseres Verständnis über Entscheidungsprozesse, Anwendungsfelder und Prioritäten in der Industrie zu gewinnen.

Bevor das eigentliche Interview begann, wurde den Experten zunächst das von Millington vorgeschlagene Kategoriensystem für KI in Spielen vorgestellt. In einem kurzen Vorgespräch konnten Rückfragen gestellt und Unklarheiten geklärt werden. Erst danach startete das eigentliche Interview. Dabei wurde bewusst darauf geachtet, weder Ergebnisse der quantitativen Analyse vorwegzunehmen noch Studierende zu befragen, sondern ausschließlich Entwickler aus der Industrie einzubeziehen.

Die Interviews sind im Anhang vollständig transkribiert. Um die Anonymität der Befragten zu wahren, werden diese im Text als Experte 1 bis Experte 7 bezeichnet. Im Folgenden werden die einzelnen Fragen des Interviews mit ihrer jeweiligen Zielsetzung näher erläutert. In den Transkriptauszügen wird mit Zeilennummern zitiert, um die Nachvollziehbarkeit zu gewährleisten.

1. **Welche weiteren Kategorien sollten berücksichtigt werden?**
   Diese Frage hat explorativen Charakter. Sie soll sicherstellen, dass das Kategoriensystem nach Millington nicht als begrenzender Rahmen verstanden wird, sondern Raum für Ergänzungen durch praktische Erfahrung bleibt. Damit wird überprüft, ob die theoretische Einteilung den Anforderungen der aktuellen Entwicklungspraxis genügt oder ob weitere algorithmische Ansätze für die Spieleindustrie relevant sind.

2. **Welche Algorithmen sind essenziell?**
   Ziel dieser Frage ist es, die aus der quantitativen Analyse hervorgehobenen Algorithmen um eine Einschätzung der praktischen Notwendigkeit zu ergänzen.

Während die Häufigkeit der Nennung lediglich statistische Relevanz anzeigt, soll durch die Experteneinschätzung ermittelt werden, welche Verfahren aus Sicht der Industrie als unverzichtbar gelten und somit eine zentrale Rolle in Entwicklungsprozessen spielen.

3. **Welche Algorithmen wurden bereits benutzt?**
   Mit dieser Frage wird die Diskrepanz zwischen theoretischer Nützlichkeit und tatsächlicher Anwendung untersucht. Sie soll aufzeigen, welche Verfahren nicht nur bekannt sind, sondern auch in realen Projekten implementiert wurden. Damit wird die zweite Teilfrage direkt adressiert, indem die praktische Umsetzung sichtbar gemacht wird.

4. **Wann und wo wurden diese Algorithmen benutzt?**
   Diese Frage dient der Kontextualisierung der Anwendung. Durch die Nennung konkreter Projekte, Genres oder Produktionsumgebungen können die Einsatzbedingungen und die Bedeutung bestimmter Algorithmen genauer eingeordnet werden. Ziel ist es, die quantitativen Ergebnisse, um praxisnahe Fallbeispiele zu erweitern und Muster im industriellen Einsatz aufzuzeigen.

Die gewählte Struktur der Interviewfragen stellt sicher, dass die quantitativen Ergebnisse nicht nur bestätigt, sondern zugleich durch konkrete Praxiserfahrungen ergänzt werden. Damit liefern die Experteninterviews eine vertiefende Erklärung, wie Algorithmen tatsächlich eingesetzt werden, und tragen dazu bei, die statistischen Befunde der quantitativen Analyse um anwendungsorientierte Perspektiven aus der Industrie zu erweitern.

4.1 Welche weiteren Kategorien berücksichtigt werden sollten

Im Rahmen der Experteninterviews wurde die bestehende AI-Kategorisierung von Millington kritisch reflektiert. Mehrere Interviewpartner gaben an, dass sie über die präsentierten Kategorien hinaus weitere Ergänzungen als sinnvoll erachten. Gleichzeitig zeigte sich, dass die vorgelegte Einteilung von vielen Experten als teils verwirrend empfunden wurde, insbesondere in Bezug auf die Kategorien Board Games und Movement. Mehrfach wurde hinterfragt, inwiefern diese Bezeichnungen tatsächlich sinnvoll und praxisnah seien oder ob sie nicht zu abstrakt gewählt wurden.

Sorting- und Validierungsalgorithmen

Ein weiterer Aspekt, der von Experte 2 hervorgehoben wurde, sind grundlegende Sorting-Algorithmen zur Ordnung von Datenlisten sowie Validation-Algorithmen für die Kommunikation zwischen Client und Server. Letztere seien besonders wichtig, um sicherzustellen, dass Treffer und Aktionen im Netzwerkspiel korrekt und fair validiert werden.[1] Zwar behandelt Millington in seinem Werk Heuristiken als Teilgebiet, doch lässt sich aus dem Expertenfeedback ableiten, dass ein eigenes Kapitel zu Heuristik und Validierung sinnvoll wäre. Dies könnte die grundlegende Bedeutung dieser Verfahren stärker hervorheben, da sie in vielen Projekten nicht als Teilbereich, sondern als eigenständige algorithmische Basis verstanden werden.

Director-AI / Scene Master

Experte 1 betonte die Relevanz einer sogenannten Director-AI (auch als Scene Master bekannt), die insbesondere in Spielen wie „Left 4 Dead“ genutzt wurde. Diese steuert die Spannungskurve, indem sie je nach Spielsituation zusätzliche Gegner, Munition oder Health-Packs einsetzt, um ein ausgewogenes Spielerlebnis zu schaffen.[2]

Game Physics

Experte 7 schlug vor, Game Physics als eigenständige Oberkategorie zu führen. Zwar könne man viele physikbasierte Verfahren unter Movement einordnen, jedoch gehe die Relevanz physikalischer Simulationen weit darüber hinaus. Gravitation, Kollisionen oder Rigid-Body-Simulationen seien in nahezu allen Spielen unverzichtbar und sollten deshalb separat betrachtet werden.[3]

Combat-Systeme

Experte 3 ergänzte die Kategorie Combat. Unter diesem Begriff fasste beispielhaft einen algorithmischen Ansatz zur dynamischen Berechnung von Hitboxen zusammen. Diese müssten in Echtzeit an die Waffenhaltung und die Spielumgebung angepasst werden, um faire Treffererkennung zu gewährleisten. Dadurch würden komplexe mathematische

---

[1] vgl Anhang P, 09.Juli 2025, Zeilen 20–40

[2] vgl. Anhang P, 09.Juli 2025, Zeilen 15–24

[3] vgl. Anhang U, 10.Juli 2025, Zeilen 34 und 52-55

Berechnungen notwendig, die in der bisherigen Kategorisierung nicht explizit enthalten sind.[1]

Keine weiteren Kategorien notwendig

Andere Experten hielten die bestehende Liste hingegen für ausreichend. So betonten Experte 4 sowie die Experten 5 und 6, dass die präsentierte Systematik alle relevanten Bereiche abdecke und keine weiteren Kategorien hinzugefügt werden müssten.[2]

4.2 Welche Algorithmen essenziell sind

Im Anschluss wurde erfragt, welche Algorithmen nach Einschätzung der Experten für die Videospielentwicklung als unverzichtbar gelten. Die Antworten verdeutlichen eine weitgehende Einigkeit, unterscheiden sich jedoch in der Gewichtung einzelner Verfahren.

Besonders häufig wurde auf Finite State Machines verwiesen. Diese gelten als zentrales Werkzeug für eine Vielzahl von Bereichen in Spielen. Sie werden sowohl für klassische KI-Aufgaben, etwa das Verhalten von Nicht-Spieler-Charakteren, als auch für generelle Spielmechaniken eingesetzt. Experte 4 betonte, dass State Machines nicht nur für Dialogsysteme und Animationen genutzt würden, sondern auch für grundlegende Steuerungsaufgaben und Game-States wie Pausen- oder Screenshot-Modi.[3] Auch in anderen Interviews wurde hervorgehoben, dass nahezu jedes Spiel in irgendeiner Form auf Zustandsmaschinen zurückgreift, da sie universell einsetzbar sind und von simplen Dialogstrukturen bis zu komplexen Verhaltenslogiken reichen.[4]

Ein zweiter Schwerpunkt lag auf Physik-Algorithmen. Diese werden in nahezu allen modernen Engines mitgeliefert und bilden die Grundlage für Kollisionserkennung, Gravitation und Interaktionen mit Objekten. Mehrere Experten äußerten, dass Physiksysteme in nahezu jedem Spiel integriert seien, auch wenn sie nicht von den Entwicklern selbst programmiert würden.[5] Der Verweis auf Physik als Basis verdeutlicht,

---

[1] vgl. Anhang R, 10.Juli 2025, Zeilen 31–56

[2] vgl. Anhang S, 10.Juli 2025, Zeile 21; Anhang T, 10.Juli 2025, Zeile 20.

[3] vgl. Anhang S, 10.Juli 2025, Zeilen 34–40

[4] vgl. Anhang U, 10.Juli 2025, Zeilen 53; Anhang R, 10.Juli 2025, Zeilen 123–126.

[5] vgl. Anhang U, 10.Juli 2025, Zeile. 59; Anhang T, 10.Juli 2025, Zeile 29

dass bestimmte algorithmische Verfahren als implizite Grundvoraussetzung für jede Form von 3D-Spielumgebung verstanden werden.

Darüber hinaus wurde Pathfinding als essenzieller Bereich genannt. Experte 1 verwies auf die enge Verbindung zu Movement und betonte, dass es sich hierbei um einen der „Basic"-Algorithmen handelt, der in vielen Genres Anwendung findet.[1] Ein anderer ergänzte, dass Pathfinding zwar nicht in jedem Spiel notwendig sei, jedoch so grundlegend sei, dass es in zahlreichen Projekten unverzichtbar werde, insbesondere in allen Anwendungen mit beweglichen Gegnern oder Agenten.[2]

Neben diesen drei wiederkehrenden Schwerpunkten nannten einzelne Befragte weitere Verfahren, die sie als essenziell ansehen, sowie grundlegende mathematische Verfahren wie Vektorrechnung und Quaternionen, die für Bewegungen, Rotationen und Transformationen in 3D-Welten unverzichtbar sind.[3] Mehrfach wurde jedoch betont, dass viele dieser Algorithmen nicht mehr eigenständig programmiert, sondern in der Regel über bestehende Game Engines oder fertige Plugins genutzt werden. So erklärte ein Experte: „Sehr viele der Sachen kannst du natürlich schon aus dem Store kaufen und in dem Fall haben wir das gekauft […]. Das ist ganz oft die Realität im Spiel. Du benutzt die Algorithmen sehr viel, aber du hast sie nicht selber angewandt."[4]

Darüber hinaus machten mehrere Experten deutlich, dass die Wahl der Algorithmen immer stark vom jeweiligen Spielkonzept abhängig ist und es daher kaum möglich ist, von universell essenziellen Verfahren auszugehen. So betonte etwa Experte 2, dass viele Algorithmen zwar grundlegend bekannt sein sollten, er in der tatsächlichen Anwendung jedoch „[…] immer eher spezielle Lösungen für spezielle Probleme" erstelle.[5]

[1] vgl. Anhang P, 09.Juli 2025, Zeilen 48-50

[2] vgl. Anhang Q, 10.Juli 2025, Zeilen 38-40

[3] vgl. Anhang R, 10.Juli 2025, Zeilen 62-37

[4] Siehe Anhang R, 10.Juli 2025, Zeilen 84-87

[5] Siehe Anhang Q, 10.Juli 2025, Zeile 71

Auch Experte 5 und 6 stellten klar, dass Kategorien wie Pathfinding oder Physics stark variieren und von den Spielmechaniken abhängen.[1,2] Ebenso zeigt Experteninterview 7, dass selbst zentrale Verfahren wie State Machines oder Physics in jedem Projekt unterschiedliche Rollen einnehmen und ihr Einsatz nicht verallgemeinert werden kann.[3]

.Zusammenfassend lässt sich festhalten, dass insbesondere drei algorithmische Bereiche von den Experten wiederholt dennoch als essenziell hervorgehoben wurden: State Machines, Physiksysteme und Pathfinding. Während die Gewichtung je nach Erfahrungsschwerpunkt der Befragten variiert, zeigt sich eine klare Tendenz dahingehend, dass diese Verfahren in der Spieleentwicklung als grundlegendes Handwerkszeug betrachtet werden. Ergänzend wurden einzelne weiterführende Kategorien genannt, die je nach Projekttyp zusätzlich Relevanz entfalten können, darunter weitere Decision-Making-Algorithmen, Heuristiken und mathematische Grundlagen.

4.3 Welche Algorithmen wurden bereits benutzt wurden

Die befragten Experten haben im Rahmen ihrer Projekte unterschiedliche Algorithmen bereits praktisch eingesetzt. Dabei zeigt sich, dass sowohl klassische Verfahren aus der KI-Programmierung als auch spezifische Lösungen für Gameplay-Mechaniken eine Rolle spielen.

Ein Beispiel liefert Experte 1, der in einem eigenen Projekt mit einer sogenannten „Director-AI“ gearbeitet hat. Ziel war es, die Spannung (Tension) im Spielverlauf dynamisch anzupassen. Dazu wurden Parameter wie Lebenspunkte, Munition und die Anzahl der Gegner ausgewertet und genutzt, um die Schwierigkeit zu regulieren. So konnte die KI beispielsweise zusätzliche Gegner spawnen oder Health-Packs bereitstellen, um das Spielerlebnis auszubalancieren.[4]

.

---

[1] vgl. Anhang T, 10.Juli 2025, Zeile 31

[2] Dabei sei darauf zu achten, dass die meisten Entwickler keinen Zusammenhang zwischen Genre und Algorithmus erwähnen, sondern viel eher zwischen dem spezifischen Spiel und einem Algorithmus. Experte 9 sagt es im ergänzenden Interview sehr prägnant: „[…] du kannst auch sagen ich kann ein RTS machen, in dem du kein Pathfinding brauchst […]“ Anhang W, 21. August 2025, Zeilen 23-24

[3] vgl. Anhang U, 10.Juli 2025, Zeilen 59-65

[4] vgl. Anhang P, 09.Juli 2025, Zeilen 63-70

Experte 2 nennt als genutzte Verfahren insbesondere Pathfinding-Algorithmen wie A*, die er sowohl im Studium als auch im beruflichen Kontext eingesetzt hat. Darüber hinaus erwähnt er Procedural Content Generation, beispielsweise für die Erstellung von Heightmaps zur Geländegenerierung. Ein weiterer von ihm angewandter Bereich sind Algorithmen zur Hit Validation im Client-Server-Kontext, mit denen Treffer im Netzwerkspiel korrekt überprüft werden.[1]

.

Auch Experte 3 hat bereits mehrere Algorithmen verwendet. Dazu zählen dynamische Berechnungen von Hitboxen für Waffen in einer isometrischen Spielwelt, die sich in Echtzeit an Ausrichtung und Länge der Waffe anpassen. Außerdem setzt er Decision-Making-Verfahren in Form von State Machines und Decision Trees ein, die über Plugins integriert wurden. Diese ermöglichen eine flexible Steuerung von NPC-Verhalten durch logische Verknüpfungen von Bedingungen.[2]

.

Experte 4 berichtet von der Nutzung klassischer KI-Algorithmen wie A*-Pathfinding und selbst entwickelten Behaviour Trees. Zudem hat er Finite State Machines eingesetzt, sowohl für die Gegner-KI als auch für andere Systeme, etwa die Steuerung von Spielzuständen. Darüber hinaus nutzte er Prozedurale Generierung, konkret für die Landschaftsgestaltung durch Terrain-Algorithmen.[3]

.

Ein praxisnahes Bild zeichnen auch Experte 5 und 6. Sie haben unter anderem Navmesh-Pathfinding, Hinderniserkennung und Behavior Trees für Entscheidungsfindung in der Gegner-KI verwendet. Darüber hinaus nennen sie physikbasierte Verfahren wie Gravitation und Rigid-Body-Simulationen, die in Engines wie Unity oder Unreal integriert sind. Auch systemische Algorithmen wie Game Manager oder Menüsteuerungen spielen für sie eine Rolle.[4]

.

Schließlich hebt Experte 7 den Einsatz von State Machines hervor, die in seinem Projekt sowohl für Animationen als auch für Dialogsysteme genutzt wurden. Zusätzlich nennt er Physics-Algorithmen, die für Kollisionen und Gravitation im 3D-Raum verantwortlich sind.

---

[1] vgl. Anhang Q, 10.Juli 2025, Zeilen 92-104

[2] vgl. Anhang R, 10.Juli 2025, Zeilen 45-52 und 83-98

[3] vgl. Anhang S, 10.Juli 2025, Zeilen 44-46 und 62-64

[4] vgl. Anhang T, 10.Juli 2025, Zeilen 49-77

Als Sonderfall beschreibt er die Integration eines externen Large-Language-Sprachmodells (LLM) über eine API, welches algorithmisch in die Spielmechanik eingebunden wurde. Dazu wurde erwähnt, dass dies ein besonderes Verfahren ist, welches die wenigsten Spiele nutzen.[1]

Zusammenfassend zeigt sich, dass die Experten vor allem auf Pathfinding (A*), Decision-Making (State Machines, Behavior Trees), Director-AI, Procedural Content Generation, Physikalgorithmen sowie spezielle Gameplay-Lösungen wie Hit Validation oder dynamische Hitboxen zurückgegriffen haben. Diese Bandbreite verdeutlicht die praktische Relevanz sowohl etablierter als auch projektspezifisch entwickelter Algorithmen in der Spieleentwicklung.

4.4 Wann und wo wurden diese Algorithmen benutzt wurden

Die Experteninterviews verdeutlichen, dass der Einsatz von Algorithmen im Kontext der konkreten Spielsituation und der Entwicklungsressourcen erfolgt. Es lassen sich drei Hauptmuster erkennen: die Nutzung integrierter Engine-Funktionalitäten, die Implementierung über Plugins, sowie die Entwicklung individueller Lösungen.

Physiksysteme wie Gravitation, Kollisionserkennung und Rigid-Body-Simulationen werden in nahezu allen Genres über die Grundfunktionen von Unity und Unreal Engine genutzt. Sie kommen in Situationen zum Einsatz, in denen die physische Glaubwürdigkeit des Spiels gewährleistet werden muss, etwa beim Zusammenstoß von Objekten, beim Springen eines Charakters oder beim Abrollen eines Projektils. Da diese Funktionen fest in die Engines integriert sind, werden sie in der Praxis selten selbst programmiert, sondern lediglich konfiguriert und in die Spielszenen eingebunden.[2]

Pathfinding-Algorithmen zeigen eine deutlich größere Abhängigkeit von der Spielsituation. In einfachen Szenarien, etwa Gegnerbewegungen in linearen Levels, wird meist das standardisierte Navmesh-System der Engine genutzt. In komplexeren Umgebungen, wie in Echtzeitstrategiespielen mit vielen Einheiten, müssen jedoch Erweiterungen programmiert werden, um etwa Gruppenbewegungen oder Hindernisverhalten effizient abzubilden. Diese

[1] vgl. Anhang U, 10.Juli 2025, Zeilen 59-65 und Zeile 63

[2] vgl. Anhang Q, 10. Juli 2025, Zeilen 92–104

Anpassungen sind notwendig, um glaubwürdige Bewegungen in dynamischen Spielsituationen zu ermöglichen.

Decision-Making-Verfahren wie Finite State Machines oder Behavior Trees werden sowohl über Plugins als auch über individuelle Implementierungen eingebunden. In der Praxis sind sie entscheidend für Situationen, in denen NPCs auf Spielereingaben reagieren müssen, etwa Gegner, die zwischen Angriff, Flucht oder Deckung wählen, oder NPCs, die abhängig vom Spielfortschritt unterschiedliche Dialoge anbieten. Während Engines und Stores häufig vorgefertigte Behavior-Tree-Tools bereitstellen, müssen Entwickler diese Systeme projektspezifisch konfigurieren und erweitern, um sie in konkrete Spielsituationen einzubetten.[1] Ein Beispiel dafür ist die Implementierung einer Director-AI, die in einem Horror-Shooter eingesetzt wurde, um die Spannungskurve dynamisch zu steuern. Diese spezifisch entwickelte Lösung griff in Echtzeit auf Variablen wie Lebenspunkte, Munitionsvorrat oder Gegneranzahl zu und generierte darauf basierend situative Anpassungen, etwa das gezielte Spawnen zusätzlicher Gegner oder das Platzieren von Health-Packs. Hier zeigt sich, wie ein eigens entwickelter Algorithmus eng mit der Dramaturgie und den Spielsituationen verknüpft ist.[2]

Als besonders innovativ gilt die Integration eines externen Sprachmodells (LLM). Dieses wurde in einem Detektivspiel genutzt, um Dialogsituationen dynamisch zu gestalten. Statt vorab definierter Entscheidungsbäume konnten Spielende freie Texteingaben tätigen, auf die das Modell reagierte. Die Implementierung stellte damit eine klare Abkehr von klassischen Verfahren dar und zeigt, wie Algorithmen auf konkrete, hochinteraktive Spielsituationen zugeschnitten werden können.[3]

Zusammenfassend wird deutlich, dass sich die Anwendung von Algorithmen durch eine enge Verzahnung von Implementierungsmethode und Spielsituation auszeichnet. Während grundlegende Systeme wie Physik fast ausschließlich enginebasiert genutzt werden, erfordern zentrale Bereiche wie Pathfinding oder Decision Making häufig hybride Lösungen aus Plugins und individuelle Anpassungen. Besondere, dramaturgisch oder spielmechanisch komplexe Anforderungen, wie Spannungskurven, dynamische

[1] vgl. Anhang U, 10. Juli 2025, Zeilen 13–27

[2] vgl. Anhang P, 09. Juli 2025, Zeilen 80-103

[3] vgl. Anhang U, 10. Juli 2025, Zeilen 61-65

Weltgestaltung oder freie Dialogführung machen individuelle Implementierungen notwendig.

4.5 Ergebnis der Experteninteviews

Die Auswertung der geführten Experteninterviews liefert mehrere zentrale Erkenntnisse zur praktischen Bedeutung und Umsetzung von Algorithmen in der Spieleentwicklung.

Zunächst wurde deutlich, dass die von Millington vorgeschlagene Kategorisierung der Spiel-KI zwar eine sinnvolle Grundlage bildet, jedoch potenziell einer Erweiterung bedarf. Besonders hervorgehoben wurde dabei die Rolle von Heuristiken, die in der Praxis häufig auch in Bereichen Anwendung finden, die außerhalb der klassischen Kategorien liegen. Dies unterstreicht, dass heuristische Verfahren eine flexible, aber zugleich schwer eindeutig zuzuordnende Funktion im Entwicklungsprozess übernehmen und daher eigenständig betrachtet werden könnten.

Ein weiterer Diskussionspunkt betraf die mögliche Neueinteilung von Algorithmen im Bereich der Physiksysteme. Zwar wurde vereinzelt vorgeschlagen, Physik als eigenständige Kategorie zu führen, jedoch lässt sich beschließen, dass diese Einordnung nicht zwingend erforderlich ist. Physikalische Verfahren sind in modernen Engines bereits vorimplementiert und werden in der Regel lediglich angepasst. Nur in sehr seltenen Fällen entstehen eigenständig programmierte Physiksysteme, wobei Entwickler auch hier verstärkt auf bestehende Plugins zurückgreifen. Die Implementierung eigener Systeme gilt aufgrund des erheblichen Zeit- und Ressourcenaufwands als Ausnahme.

Ein zentrales Ergebnis betrifft die Nutzung vorhandener Plugins und Engine-Funktionalitäten. Die Experten betonten übereinstimmend, dass es in der Praxis selten notwendig ist, grundlegende Algorithmen neu zu entwickeln. Stattdessen werden bereits implementierte Lösungen bevorzugt, um Entwicklungszeit, Kosten und Mühen einzusparen. Entwickler wählen in der Praxis Algorithmen nicht, indem sie abstrakt überlegen, welcher Algorithmus einem Problem theoretisch zugeordnet werden könnte. Vielmehr steht stets die Lösung eines konkreten Problems im Vordergrund. Dabei ergibt sich der Einsatz eines Algorithmus häufig situativ, sei es durch den Rückgriff auf bereits vorhandene Plugins, durch die Anpassung bestehender Verfahren oder durch die Implementierung einer eigenen Lösung. Nicht immer ist den Entwicklern dabei bewusst, welche spezifische algorithmische

Methode im Hintergrund wirksam ist. Kenntnisse über vorhandene Algorithmen erleichtern zwar die Recherche und Auswahl geeigneter Ansätze, doch erfolgt der Wissensaufbau in der Regel inkrementell und erfahrungsbasiert. Entwickler arbeiten sich nicht systematisch durch eine vollständige Liste aller bekannten Algorithmen, sondern erweitern ihr Repertoire schrittweise anhand praktischer Projekte.

Bezüglich essenzieller Algorithmen bestand ein differenzierter Konsens, dass es keine universell unverzichtbaren Algorithmen gibt. Dennoch kristallisierten sich drei Hauptkategorien heraus: Decision-Making, Pathfinding und Physik. Diese wurden von fast allen Experten als praxisrelevant hervorgehoben. Konkret nannten 6 von 10 Befragten Decision Making, 5 Pathfinding und jeweils 3 Sorting bzw. Physics.[1]

Die Interviews verdeutlichten zudem, dass die konkreten Beispiele der Befragten die Vielseitigkeit von Anwendungsfällen illustrieren. Algorithmen wurden sowohl für grundlegende Mechaniken, wie das Movement, als auch für spezifische Systeme wie prozedurale Terrain-Generierung, dynamische Hitboxen oder dramaturgische Spannungskurven genutzt. Damit wird sichtbar, dass Algorithmen in sehr unterschiedlichen Kontexten eingesetzt werden und eine hohe Bandbreite an Aufgabenbereichen abdecken.
Die am häufigsten benannten Verfahren decken sich größtenteils mit den Ergebnissen der quantitativen Analyse, wobei insbesondere Movement, Decision Making und Pathfinding in beiden Untersuchungen hohe Relevanz aufweisen. Gleichwohl ist diese Beobachtung mit Vorsicht zu bewerten, da die geringe Zahl an Interviews keine verlässliche statistische Aussage erlaubt und lediglich Tendenzen sichtbar macht.

Abschließend lässt sich festhalten, dass die Experten den situativen Charakter von Algorithmen betonen. Ihre Auswahl und Implementierung sind stets stark vom spezifischen Problem abhängig, das gelöst werden soll. Viele Verfahren werden deshalb nicht wiederholt in identischer Form genutzt, sondern gezielt an den jeweiligen Anwendungsfall angepasst. Algorithmen erscheinen somit weniger als standardisierte, universell übertragbare Werkzeuge, sondern vielmehr als flexible Problemlösungsansätze, die ihre Relevanz aus dem konkreten Kontext der Spieleentwicklung beziehen.

[1] Die Zahl von maximal zehn Experten ergibt sich aus den ergänzenden Gamescom Experteninterviews aus Abschnitt 5.4. Vollständige Tabellen sind in Anhang Y und Z zu finden.

5 Überprüfung Und Bewertung der Rechercheergebnisse

5.1 Validität und Reliabilität der Ergebnisse

Die quantitative Analyse dieser Arbeit weist sowohl in Bezug auf ihre Validität als auch auf ihre Reliabilität bestimmte Stärken und Einschränkungen auf. Die verwendete Datengrundlage, bestehend aus den Game Awards Nominierungen und Artikeln der Fachzeitschrift Game Developer, ist systematisch erhoben, bleibt jedoch auf ein spezifisches Segment beschränkt. Berücksichtigt wurden vor allem nominierte Spiele, die überwiegend aus dem Bereich der AAA-Produktionen stammen. Dies führt zu einer möglichen Verzerrung, da ausschließlich Spiele mit hoher Sichtbarkeit und Medienpräsenz erfasst wurden. Damit ist die Validität insofern eingeschränkt, als nicht alle Spiele oder Genres gleichermaßen repräsentiert sind.

Hinsichtlich der Reliabilität kann festgehalten werden, dass die automatisierte Stichwortsuche konsistente Ergebnisse liefert, jedoch die Gefahr von Fehlinterpretationen birgt. Mehrdeutigkeiten in Begriffen oder Verluste durch den Kontext können dazu führen, dass einzelne Zuordnungen ungenau ausfallen. Die eingesetzten Filter- und Bereinigungsschritte erhöhen zwar die Zuverlässigkeit der Ergebnisse, bergen jedoch gleichzeitig das Risiko, dass relevante Daten versehentlich ausgeschlossen wurden.

Die qualitativen Interviews zeichnen sich durch eine hohe Praxisnähe aus, da die Befragten aktive Entwickler sind und somit authentische Einblicke in die Anwendung von Algorithmen geben konnten. Dennoch ist die Validität insofern eingeschränkt, als die Auswahl der Interviewpartner zwar zufällig erfolgte, jedoch nicht repräsentativ für die gesamte Branche ist. Hinzu kommt die Möglichkeit subjektiver Verzerrungen, die aus individuellen Erfahrungen oder Erinnerungseffekten der Befragten resultieren können.

In Bezug auf die Reliabilität wurde durch die Durchführung der Interviews nach einem einheitlichen Leitfaden ein gewisser Grad an Vergleichbarkeit gewährleistet. Allerdings reduziert die begrenzte Anzahl an Interviews die statistische Belastbarkeit der Ergebnisse.

5.2 Methodische Einschränkungen und Verzerrungen

Trotz der systematischen Vorgehensweise weist die Arbeit mehrere methodische Einschränkungen und potenzielle Verzerrungen auf, die bei der Interpretation der Ergebnisse berücksichtigt werden müssen.

Ein wesentlicher Aspekt betrifft die Auswahl der Datenbasis. Die quantitative Untersuchung stützt sich auf die Nominierungen der Game Awards sowie auf Artikel aus dem Fachmagazin Game Developer. Diese Eingrenzung führt zwangsläufig zu einem Survivorship Bias, da nur Spiele betrachtet werden, die bereits eine gewisse Sichtbarkeit und Anerkennung erlangt haben. Titel, die möglicherweise innovative algorithmische Ansätze enthalten, jedoch nicht nominiert oder publizistisch hervorgehoben wurden, bleiben unberücksichtigt. Dies begünstigt eine Verzerrung zugunsten etablierter AAA-Produktionen und kann die Vielfalt der tatsächlich eingesetzten Algorithmen einschränken.

Ein zweiter methodischer Aspekt betrifft die Definition von Relevanz. Der Umsatz der Spiele wurde bewusst nicht als Metrik herangezogen, um eine einseitige ökonomische Bewertung zu vermeiden. Gleichwohl stellt dies eine methodische Begrenzung dar, da sich der Einfluss von Algorithmen auf wirtschaftliche Faktoren beispielsweise Entwicklungskosten oder potenzielle Umsatzsteigerungen durch bestimmte KI-Verfahren nicht erfassen lässt. Eine weiterführende Untersuchung könnte gezielt analysieren, welche Algorithmen in welchem Maße Kosten verursachen oder ökonomischen Mehrwert generieren.

Darüber hinaus bestehen Einschränkungen in Bezug auf die Durchführung und Auswertung der Interviews. Zwar liefern die Befragungen wertvolle qualitative Einblicke, doch sind diese durch die begrenzte Zahl an Teilnehmern in ihrer Repräsentativität eingeschränkt. Hinzu kommt, dass sich die Auswahl ausschließlich auf Personen beschränkt, die die GDD besucht haben und dabei nur deutsche Entwickler berücksichtigt wurden. Die Perspektive anderer internationaler Entwickler bleibt damit unberücksichtigt, was die Generalisierbarkeit zusätzlich einschränkt. Zudem wurde die Auswahl gezielt auf Programmierer fokussiert, ohne die unterschiedlichen Spezialisierungen innerhalb dieser Berufsgruppe systematisch zu erfassen. Dadurch ist es möglich, dass bestimmte Sichtweisen überrepräsentiert sind, während andere relevante Erfahrungsbereiche nicht in die Untersuchung eingeflossen sind. Subjektive Verzerrungen durch individuelle Erfahrungen oder persönliche Vorlieben der Befragten lassen sich ebenfalls nicht ausschließen.

5.3 Einordnung der quantitativen Ergebnisse mithilfe qualitativer Interviews

Die qualitativen Interviews tragen wesentlich zur Einordnung der quantitativen Analyse bei und ergänzen diese um praxisorientierte Perspektiven. Während die quantitative Untersuchung in erster Linie die Häufigkeit bestimmter Algorithmen in den analysierten Quellen widerspiegelt, verdeutlichen die Interviews, warum spezifische Verfahren in der Entwicklungspraxis dominieren und welche Faktoren ihre Relevanz bestimmen.

Unter Berücksichtigung der methodischen Verzerrungen ergibt die quantitative Analyse folgende Rangfolge:

1. Movement-Algorithmen,
2. Decision-Making-Verfahren
3. Pathfinding.

Darüber hinaus zeigt sich ein wachsender Trend zur Verwendung von Learning-Algorithmen, die an Popularität gewinnen und zunehmend in Entwicklungsprozesse integriert werden.

Die qualitative Analyse kommt zu einer leicht abweichenden Bewertung. Nach Einschätzung der befragten Experten ergibt sich folgende Relevanzreihung:

1. Decision-Making
2. Pathfinding
3. Physiksysteme (Movement)

Hervorzuheben ist, dass physikalische Algorithmen an Bedeutung verlieren, da sie in modernen Engines bereits umfassend vorimplementiert sind und individuelle Implementierungen nur noch selten vorgenommen werden.

Die Gegenüberstellung der quantitativen und qualitativen Ergebnisse zeigt ein konsistentes Muster. Decision-Making, Pathfinding und Movement (einschließlich physikalischer Verfahren nach Millington) erscheinen in beiden Ansätzen unter den drei wichtigsten Kategorien. Zwar variiert die Rangfolge, doch die wiederholte Präsenz dieser Bereiche unterstreicht ihre zentrale Bedeutung für die Spieleentwicklung.

Movement wird quantitativ stark hervorgehoben, während es in den Interviews weniger Gewicht erhält, da entsprechende Funktionen häufig durch Engines vorgegeben sind.

Decision-Making und Pathfinding werden hingegen in beiden Analysen deutlich als Schlüsselfaktoren bestätigt. Ergänzend ist zu beobachten, dass Learning-Algorithmen zunehmend an Relevanz gewinnen, aktuell jedoch noch eine untergeordnete Rolle spielen.

5.4 Ergänzende Experteninterviews auf der Gamescom

Um den Datensatz zu erweitern und die Experteninterviews auf den GDD 2025 zu evaluieren wurden auf der Gamescom 2025 drei weitere Experteninterviews geführt.
Die ergänzenden Experteninterviews 8 bis 10 wurden im Rahmen auf Gamescom 2025 in der Indie Arena Booth durchgeführt. Vorbereitung und Fragestellungen entsprachen exakt den zuvor auf den GDD geführten Gesprächen, um methodische Vergleichbarkeit sicherzustellen.

Die Aussagen der Befragten bestätigen in wesentlichen Punkten die Ergebnisse der GDD-Interviews. Besonders Pathfinding und Decision Making wurden erneut als zentrale Verfahren benannt. So betonte Experte 8: “Pathfinding is extremely widespread […] Pathfinding is therefore at the top. “.[1] Auch Experte 10 erklärte, dass Klassiker wie die A*-Suche „essenziell sind, wenn man mit Nodes oder Tiles arbeitet“.[2]

Eine deutliche Ergänzung stellt die Hervorhebung von Sortieralgorithmen dar. Experte 8 bezeichnete sie als „das Brot-und-Butter des Programmierens“.[3] Auch Experte 9 hob Sortieren als eigenständige Kategorie hervor und schilderte ein praxisnahes Beispiel: Obwohl Bubblesort in der Lehre als ineffizient gilt, erwies er sich in einem MMO als optimale Lösung, da Spielerstatistiken „immer dicht dran“ waren und sich lokal effizient korrigieren ließen.[4] Dieses Beispiel verdeutlicht, dass die Wahl von Algorithmen in der Praxis stark von den jeweiligen Rahmenbedingungen abhängt. Experte 10 bestätigte die Relevanz heuristischer Verfahren, die Inhalte oder Gegnerentscheidungen an den Spielkontext anpassen.[5]

[1] Experte 8, 21. August 2025, Zeilen 30–33

[2] Experte 10, 21. August 2025, Zeile 62

[3] Experte 8, 21. August 2025, Zeile 41

[4] vgl. Experte 9, 21. August 2025, Zeilen 44–55

[5] vgl. Experte 10, 21. August 2025, Zeile. 55

Übereinstimmend äußerten alle Befragten, dass die Auswahl geeigneter Algorithmen immer vom konkreten Problem ausgeht. Experte 9 formulierte dies prägnant: „Du solltest überlegen, was möchte ich erreichen oder welches Problem habe ich? Und dann schaust du das Problem genau an […] Kategorien kennen ist gut […], aber immer erstmal suchen und dann nicht eine Standardlösung nehmen […]“.[1]

Zusammenfassend ergibt sich ein klarer Konsens: Pathfinding und Decision Making bilden die Grundlage vieler Spielsysteme. Sortieralgorithmen, bisher kaum thematisiert, wurden hier neu als universelle Basiswerkzeuge hervorgehoben. Prozedurale Content-Generierung ist ein weiteres zentrales Feld, dessen Einsatz stark vom Problem abhängt. Die Experteninterviews auf der Gamescom unterstreichen damit die Befunde der vorangegangenen Gespräche und akzentuieren zugleich, dass nicht starre Standards, sondern die problemorientierte Anwendung bekannter algorithmischer Prinzipien den Kern der Spieleentwicklung bildet.

[1] Experte 9, 21. August 2025, Zeile 33 und Zeilen 59-60

6 Ansätze für weiterführende Arbeiten

Aus den Methoden und Ergebnissen ergeben sich mehrere Ansätze für zukünftige Arbeiten. Besonders relevant ist die Erweiterung der Datengrundlage. Anstelle der Beschränkung auf Game Awards-Nominierungen und Fachartikel sollten auch Entwicklerblogs, Postmortems, Indie-Projekte oder internationale Studien einbezogen werden. Bisher fehlte jedoch eine verlässliche Datenbank, weshalb eine solche Sammlung entweder manuell erfolgen oder durch den Aufbau einer speziellen Algorithmen-Videospiel-Datenbank systematisch erschlossen werden müsste.

Darüber hinaus könnten die ökonomischen Dimensionen von Algorithmen in den Fokus rücken. Der Umsatz wurde in dieser Untersuchung bewusst nicht berücksichtigt, eröffnet jedoch ein Forschungsfeld, das den Zusammenhang zwischen eingesetzten Algorithmen, Entwicklungskosten und wirtschaftlichem Erfolg näher untersucht. Eine systematische Erhebung dieser Faktoren könnte aufzeigen, welche Verfahren nicht nur technisch, sondern auch betriebswirtschaftlich besonders relevant sind.

Schließlich erscheint es lohnend, die dynamische Entwicklung neuer Verfahren, insbesondere von Learning-Algorithmen, intensiver zu beobachten. Obwohl diese derzeit noch eine untergeordnete Rolle spielen, weisen sie auf ein wachsendes Potenzial hin, das zukünftige Forschungsarbeiten stärker berücksichtigen sollten.

7 Fazit

Ziel der Arbeit war es, relevante Algorithmen in der Videospielentwicklung zu identifizieren, ihren Bezug zu Genres zu prüfen und ihre Umsetzung in Game Engines zu analysieren. Grundlage war ein Mixed-Methods-Ansatz aus quantitativer Analyse und Experteninterviews.

Die Leitfrage nach den wichtigsten Algorithmen zeigt: Movement, Decision Making und Pathfinding dominieren, während Learning-Algorithmen an Bedeutung gewinnen. Sie bilden die Basis vieler Spiele, auch wenn Experten betonen, dass es keine universell essenziellen Algorithmen gibt, sondern nur solche, die häufig vorkommen.

Zur ersten Teilfrage, dem Zusammenhang zwischen Algorithmen und Genres, ergab die Analyse keinen statistisch signifikanten Zusammenhang. Algorithmen sind weitgehend genreübergreifend nutzbar. Die Interviews verdeutlichen, dass sie eher auf Spielebene als auf Genre-Ebene relevant werden. Weitere Forschung mit größeren Datensätzen wäre sinnvoll.

Die zweite Teilfrage nach der praktischen Anwendung zeigt: Viele Verfahren sind in Engines wie Unity oder Unreal vorimplementiert, etwa Navmesh-Pathfinding, Physiksimulationen oder Behavior Trees. Der Einsatz reicht von Standardlösungen über zugekaufte Packages bis hin zu spezifischen Eigenentwicklungen. Entscheidend ist nicht allein das Vorhandensein eines Algorithmus, sondern die Tiefe seiner Implementierung.

Insgesamt bestätigt die Arbeit, dass Algorithmen grundlegende Werkzeuge der Spieleentwicklung darstellen, deren Bedeutung jedoch stark vom Projektkontext abhängt. Die Forschung konnte zeigen, dass bestehende Kategorisierungen wie bei Millington zwar Orientierung bieten, jedoch in Teilen zu abstrakt bleiben und durch eine eigene Heuristik oder Sortieralgorithmen Kategorie ergänzt werden sollten. Die Ergebnisse tragen dazu bei, die Lücke zwischen theoretischer Einordnung und praktischer Anwendung zu verringern, weisen jedoch auf die Notwendigkeit breiter angelegter Forschung hin.

Literaturverzeichnis

Astle, A. (September 2023). *Pocketgamer.* Abgerufen am 28. Juni 2025 von https://www.pocketgamer.biz/71-of-the-biggest-mobile-games-use-a-unity-sdk/

Bossom, A., & Dunning, B. (2015). *Video Games: An Introduction to the Industry.* Fairchild Books.

Farell, N. (21. November 2024). *The Game Awards 2024: All Nominees For Each Category*. Abgerufen am 3. Juli 2025 von https://www.turtlebeach.com/blog/the-game-awards-2024-all-nominees-for-each-category?srsltid=AfmBOooAIoHLKSn_cAF_cAOkr_IHKGgBD5zDrSQzOGvjc4FbNbsvQmgB

Godot. (o.J.). *Godot.* Abgerufen am 29. Juni 2025 von https://godotengine.org/

Harte, C. (14. November 2022). *The Full List of the 2022 Game Awards Nominees*. Abgerufen am 3. Juli 2025 von https://gameinformer.com/2022/11/14/the-full-list-of-the-2022-game-awards-nominees

Holfeld, J. (10. Januar 2024). *On the relevance of the Godot Engine in the indie game development industry.* (C. University, Hrsg.) Abgerufen am 29. Juni 2025 von https://arxiv.org/abs/2401.01909

INNOVECS Games. (9. Juli 2025). *Wie lange dauert die Entwicklung eines AAA-Spiels? Ein umfassender Überblick*. Abgerufen am 25. Juli 2025 von https://www.innovecsgames.com/blog/how-long-does-it-take-to-make-an-aaa-game/#:~:text=Most%20AAA%20games%20require%20years,time%20and%20effort%20to%20achieve.

Kelbert, P., Siebert, J. D., & Jöckl, L. (12. Dezember 2023). *Was sind Large Language Models? Und was ist bei der Nutzung von KI-Sprachmodellen zu beachten?* Abgerufen am 23. August 2025 von https://www.iese.fraunhofer.de/blog/large-language-models-ki-sprachmodelle/

Kiselev, M. (19. September 2024). *Real reasons (not) to build custom game engines in 2024*. Abgerufen am 30. Juni 2025 von https://www.gamedeveloper.com/programming/real-reasons-not-to-build-custom-game-engines-in-2024

Makuch, E. (1. Dezember 2014). *2014 Game Awards Nominees Announced*. Abgerufen am 5. Juli 2025 von https://www.gamespot.com/articles/2014-game-awards-nominees-announced/1100-6423734/

Makuch, E. (1. Dezember 2016). *All the 2016 Game Awards Nominees*. Abgerufen am 4. Juli 2025 von https://www.gamespot.com/articles/all-the-2016-game-awards-nominees/1100-6445481/

Makuch, E. (14. November 2017). *All The 2017 Game Awards Nominees*. Abgerufen am 4. Juli 2025 von https://www.gamespot.com/articles/all-the-2017-game-awards-nominees/1100-6454944/

Mcwhertor, M. (11. Dezember 2024). *How The Game Awards voting works*. Abgerufen am 3. Juli 2025 von https://www.polygon.com/explained/492637/how-the-game-awards-voting-works/

Millington, I. (2019). *AI for Games* (3. Ausg.). CRC Press.

Ngulube, P. (2021). *Handbook of Research on Mixed Methods Research in Information Science.* IGI Global.

Oxford University Press. (o.J). *Ocford Learner's Dictionaries / What is the CERF*. Abgerufen am 27. Juli 2025 von https://www.oxfordlearnersdictionaries.com/about/wordlists/cefr

Oxford University Press. (o.J.). *Oxford Learner's Dictionaries*. Abgerufen am 15. Juli 2025 von https://www.oxfordlearnersdictionaries.com/about/wordlists/oxford3000-5000

Park, G. (20. November 2020). *Here are the nominees for The Game Awards 2020*. Abgerufen am 3. Juli 2025 von https://www.washingtonpost.com/video-games/2020/11/18/game-awards-2020-nominees/

Plant, L. (13. Dezember 2019). *The Game Awards 2019: All the Nominees*. Abgerufen am 4. Juli 2025 von https://www.ign.com/articles/2019/11/19/the-game-awards-2019-all-the-nominees

Plotnek, J. (05. Mai 2025). *Keewano.* Abgerufen am 28. Juni 2025 von https://keewano.com/blog/best-game-engines-mobile-game-development/#:~:text=Developed%20by%20YoYo%20Games%20with,may%20struggle%20with%20advanced%20features.

Rabin, S. (2002). *AI Game Programming Wisdom.* Charles River Media, Inc.

Rabin, S. (2003). *AI Game Programming Wisdom 2.* Charles River Media, Inc.

Rabin, S. (2006). *AI Game Programming Wisdom 3.* Charles River Media, Inc.

Rabin, S. (2008). *AI Game Programming Wisdom 4.* Charles River Media, Inc.

Rabin, S. (2013). *Game AI Pro.* CRC Press.

Rabin, S. (2015). *Game AI Pro 2.* CRC Press.

Rabin, S. (2017). *Game AI Pro 3.* CRC Press.

Rabin, S. (2021). *Game AI Pro 4*. Von https://www.gameaipro.com/ abgerufen

Rabin, S. (o.J.). *Game AI Pro*. Abgerufen am 22. August 2025 von https://www.gameaipro.com/

RAWG. (o.J.). *RAWG API Dokumentation*. Abgerufen am 16. Juli 2025 von https://api.rawg.io/docs/

Reeves, B. (13. November 2018). *The Game Awards Announces 2018 Nominees*. Abgerufen am 4. Juli 2025 von https://gameinformer.com/2018/11/13/the-game-awards-announces-2018-nominees

Roberts, P. (2022). *Artificial Intelligence in Games.* CRC Press.

Sarkar, S. (13. November 2015). *Here are the nominees for The Game Awards 2015*. Abgerufen am 5. Juli 2025 von https://www.polygon.com/2015/11/13/9728874/the-game-awards-2015-nominees/

Stewart, M. (16. November 2021). *Here Are The Nominees For The Game Awards 2021*. Abgerufen am 3. Juli 2025 von https://gameinformer.com/2021/11/16/here-are-the-nominees-for-the-game-awards-2021

The Game Awards. (2025). *The Game Awards / about*. Abgerufen am 23. Juli 2025 von https://thegameawards.com/about

Turtle Beach. (13. November 2023). *The Game Awards 2023 All Nominees*. Abgerufen am 3. Juli 2025 von https://www.turtlebeach.com/blog/the-game-awards-2023-all-nominees

Video Game Insights Ltd. (3. Februar 2025). *The Big Game Engines Report of 2025.* Abgerufen am 27. Juni 2025 von https://vginsights.com/assets/reports/The_Big_Game_Engines_Report_of_2025.pdf

Zhiyanov, A. (28. November 2021). *GitHub*. Abgerufen am 23. Juli 2025 von https://github.com/nalgeon/words/blob/main/data/oxford-5k.csv

Zukowski, C. (16. Juli 2024). *What games are selling on Steam: The Q2 Report*. Abgerufen am 30. Juni 2025 von https://howtomarketagame.com/2024/07/16/what-games-are-selling-q2-2024/

Übersicht verwendeter Hilfsmittel

Hilfsmittel	Anwendungsfall	Quelle
Beautiful Soup	Zum Parsen von HTML-Webseiten	https://pypi.org/project/beautifulsoup4/
Python Requests Module	Zum Ansprechen von Webseiten	https://www.w3schools.com/python/module_requests.asp
ChatGPT	Zum Paraphrasieren und als Unterstützung zur Erstellung von Python-Skripten	https://chatgpt.com
Microsoft Excel	Zur Erstellung eigener Grafiken	https://www.microsoft.com/de-de/microsoft-365/excel?market=de
Microsoft Word	Zum Schreiben der Arbeit	microsoft.com/de-de/microsoft-365/word
Visual Studio Code	Zum Ausführen der Python Skripte	https://code.visualstudio.com/
Python	Zum Datensammeln & Verarbeiten	https://www.python.org/downloads/

Anhang

Anhang A: Game Developer Links Programming Filter Suche

```python
import csv
import requests
from bs4 import BeautifulSoup
import urllib.parse
import time

# Basis-URL für die Suche
BASIS_SUCH_URL = "https://www.gamedeveloper.com/search"

# Name der Ausgabedatei
AUSGABE_DATEI = "Gamedeveloper_Links_mit_Programming_Filter.csv"

# Name der Eingabedatei mit allen Spielen
EINGABE_DATEI = "Liste_aller_nominierten_Spiele.csv"

# Header, damit die Anfrage wie ein Browser aussieht
KOPFZEILEN = {"User-Agent": "Mozilla/5.0"}

# Wartezeit zwischen den Anfragen
WARTEZEIT = 1.5

# Spielname kodieren und vollständige Such-URL bauen
def erstelle_such_url(spielname):
    kodiert = urllib.parse.quote_plus(spielname)
    return f"{BASIS_SUCH_URL}?q={kodiert}&terms=%2Fprogramming"

# Spiel auf der Webseite suchen und Links zurückgeben
def suche_spiel(spielname):
    url = erstelle_such_url(spielname)
    try:
        antwort = requests.get(url, headers=KOPFZEILEN, timeout=10)
        suppe = BeautifulSoup(antwort.text, 'html.parser')

        # Prüfen, ob keine Ergebnisse gefunden wurden
        keine = suppe.find(string=lambda t: "Keine Ergebnise gefunden für: " in t if t else False)
        if keine:
            return '✗', []

        # Liste für Links erstellen
        links = []
        gesehen = set()

        # Alle passenden Links sammeln
        for link_tag in suppe.select("a.ListPreview-Title[href]"):
            href = link_tag['href']
            href = urllib.parse.urljoin("https://www.gamedeveloper.com", href)
            if href not in gesehen:
                links.append(href)
                gesehen.add(href)

        return '✓', links

    except Exception as fehler:
        # Fehlerfall abfangen
        print(f"Fehler bei '{spielname}': {fehler}")
```

```
        return 'X', []

def hauptfunktion():
    # Spielnamen aus Eingabedatei lesen
    print("Lese Spielliste...")
    with open(EINGABE_DATEI, newline='', encoding='utf-8') as f:
        leser = csv.reader(f)
        spiele = [zeile[0].strip() for zeile in leser if zeile]

    print(f"{len(spiele)} Spiele gefunden. Starte Suche...")

    ergebnisse = []
    gesamt_links = 0

    # Alle Spiele durchsuchen
    for nr, spiel in enumerate(spiele, start=1):
        print(f"[{nr}/{len(spiele)}] Suche nach: {spiel}")
        status, links = suche_spiel(spiel)
        print(f"Ergebnis: {status}")
        if links:
            print(f"  ↳ {len(links)} Link(s) gefunden")
            gesamt_links += len(links)
            for link in links:
                ergebnisse.append([spiel, status, link])
        else:
            print("  ↳ Keine Links gefunden")
            ergebnisse.append([spiel, status, ""])
        time.sleep(WARTEZEIT)

    # Ergebnisse in Ausgabedatei speichern
    print(f"\nSpeichere Ergebnisse in {AUSGABE_DATEI}...")
    with open(AUSGABE_DATEI, 'w', newline='', encoding='utf-8') as f:
        schreiber = csv.writer(f)
        schreiber.writerow(['Spiel', 'Ergebnis', 'Link'])
        schreiber.writerows(ergebnisse)

    # Zusammenfassung ausgeben
    print(f"Gesamtanzahl gefundener Links: {gesamt_links}")
    print("☑ Fertig!")

# Programm starten
if __name__ == "__main__":
    hauptfunktion()

```

Quellcode 1: Python Script zur Ermittlung von Links der Game Developer Webseite. Es wird ein CSV-Datei mit einer Liste aller nominierten Game Awards Spiele genommen und eine neue CSV-Datei mit den Spieletiteln und den gefundenen Webseiten erstellt.

Anhang B: Game Developer Stichwort Suche

```
import csv
import requests
from bs4 import BeautifulSoup
import time
import re

EINGABE_DATEI = "Gamedeveloper_Links_mit_Programming_Filter.csv"
AUSGABE_DATEI = "Gamedeveloper_Stichwort_Ergebnisse.csv"
HEADERS = {"User-Agent": "Mozilla/5.0"}
WARTEZEIT_ZWISCHEN_ANFRAGEN = 1.1  # sekunden

KATEGORIEN = {
    # Movement
    "Movement": [
    "Movement","Basics of Movement Algorithm", # Die Bibliothek der Stichwortliste wurde aus leserlichen Zwecken gekürzt.
    ]
}

# Funktion: Ruft den Text einer Webseite ab
def seiten_text_abrufen(url):
    try:
        antwort = requests.get(url, headers=HEADERS, timeout=10)
        if antwort.status_code != 200:
            print(f"Fehler beim Abrufen: Status {antwort.status_code}")
            return ""
        suppe = BeautifulSoup(antwort.text, 'html.parser')
        span_elemente = suppe.find_all("span", class_="ContentText ContentText_variant_bodyNormal")
        text = " ".join(span.get_text(separator=' ', strip=True) for span in span_elemente)
        return text.lower()
    except Exception as e:
        print(f"Fehler beim Abrufen von {url}: {e}")
        return ""

# Funktion: Überprüft, ob Schlüsselwörter im Text vorkommen
def schluesselwoerter_pruefen(text, schluesselwoerter):
    ergebnisse = {}
    for wort in schluesselwoerter:
        muster = re.compile(re.escape(wort), re.IGNORECASE)
        treffer = muster.search(text)
        if treffer:
            start, ende = treffer.span()
            praefix, suffix = "", ""

            i = start - 1
            while i >= 0 and text[i].isalnum():
                praefix = text[i] + praefix
                i -= 1

            i = ende
            while i < len(text) and text[i].isalnum():
                suffix += text[i]
                i += 1

            ergebnisse[wort] = f"({praefix}){wort}({suffix})" if praefix or suffix else wort
        else:
            ergebnisse[wort] = ""
    return ergebnisse
```

```
58.
59. # Hauptfunktion: Liest CSV, ruft Webseiten ab und prüft Schlüsselwörter
60. def hauptfunktion():
61.     print("Eingabe-CSV wird gelesen...")
62.     with open(EINGABE_DATEI, newline='', encoding='utf-8') as f:
63.         leser = csv.reader(f)
64.         daten = [zeile for idx, zeile in enumerate(leser) if idx > 0 and len(zeile) >= 3 and zeile[2].strip()]
65.
66.     print(f"Es werden {len(daten)} gültige Links verarbeitet...\n")
67.
68.     # Schlüsselwörter aus Kategorien sammeln
69.     alle_schluesselwoerter = []
70.     for kategorie, woerter in KATEGORIEN.items():
71.         alle_schluesselwoerter.extend([(kategorie, wort) for wort in woerter])
72.
73.     ergebnisse = []
74.     for idx, zeile in enumerate(daten, start=1):
75.         spiel, url = zeile[0].strip(), zeile[2].strip()
76.         print(f"[{idx}/{len(daten)}] Durchsuche '{spiel}' unter {url}")
77.         text = seiten_text_abrufen(url)
78.         gefundene_woerter = schluesselwoerter_pruefen(text, [wort for _, wort in alle_schluesselwoerter])
79.
80.         gefundene = [v for v in gefundene_woerter.values() if v]
81.         if gefundene:
82.             print(f"{len(gefundene)} Schlüsselwort(er) gefunden.")
83.         else:
84.             print("Keine Schlüsselwörter gefunden.")
85.
86.         zeile_out = [spiel, url] + [gefundene_woerter[wort] for _, wort in alle_schluesselwoerter]
87.         ergebnisse.append(zeile_out)
88.         time.sleep(WARTEZEIT_ZWISCHEN_ANFRAGEN)
89.
90.     print(f"\nSpeichere Ergebnisse in {AUSGABE_DATEI}...")
91.     with open(AUSGABE_DATEI, 'w', newline='', encoding='utf-8') as f:
92.         schreiber = csv.writer(f)
93.         kopfzeile = ["Spiel", "Webseite"] + [f"{kat}: {wort}" for kat, wort in alle_schluesselwoerter]
94.         schreiber.writerow(kopfzeile)
95.         schreiber.writerows(ergebnisse)
96.
97.     print("✓ Fertig!")
98.
99. if __name__ == "__main__":
100.     hauptfunktion()
101.
```

Quellcode 2: Python Skript das Stichwörter in gegebenen Links der Game Developer Webseite sucht. Als Ergebnis wird eine CSV-Datei ausgegeben.

Angang C: RAWG Genre Suche

```
import csv
import os
import time
import requests

API_SCHLUESSEL = "xxxxxxxxxxxxxxxxxx"  # <-- Hier den eigenen RAWG API Key einfügen
BASIS_URL = "https://api.rawg.io/api/games"
eingabe_datei = "Liste_aller_nominierten_Spiele.csv"
ausgabe_datei = "Liste_aller_nominierten_Spiele_mit_Genre.csv"

def suche_spiel_rawg(spielname):
    """Suche ein Spiel bei RAWG und gib dessen Genre(s) zurück."""
    print(f"\n[SUCHEN] Suche nach: {spielname}")
    parameter = {
        "key": API_SCHLUESSEL,
        "search": spielname,
        "page_size": 1  # nur das beste Ergebnis
    }
    try:
        antwort = requests.get(BASIS_URL, params=parameter, timeout=10)
    except requests.RequestException as e:
        print(f"[FEHLER] Anfrage fehlgeschlagen für '{spielname}': {e}")
        return []

    if antwort.status_code != 200:
        print(f"[FEHLER] API-Anfrage fehlgeschlagen für '{spielname}' (Status {antwort.status_code})")
        return []

    daten = antwort.json()
    if not daten.get("results"):
        print(f"[WARNUNG] Keine Ergebnisse für '{spielname}' gefunden")
        return []

    erstes_ergebnis = daten["results"][0]
    ergebnis_name = erstes_ergebnis.get("name", "Unbekannt")
    genres = [g["name"] for g in erstes_ergebnis.get("genres", [])]

    print(f"[TREFFER] RAWG lieferte: {ergebnis_name}")
    print(f"[GENRE] {', '.join(genres) if genres else 'Kein Genre gefunden'}")

    return genres

    # Prüfen ob Eingabedatei existiert
def main():
    if not os.path.exists(eingabe_datei):
        print(f"[FEHLER] Eingabedatei '{eingabe_datei}' nicht gefunden.")
        return

    # Spieleliste aus CSV einlesen
    with open(eingabe_datei, newline='', encoding='utf-8') as f:
        leser = csv.reader(f)
        spiele = [zeile[0] for zeile in leser if zeile]

    ergebnisse = []      # Liste mit Spielen und deren Genres
    alle_genres = set()  # Menge aller gefundenen Genres

    # Jedes Spiel verarbeiten
    for index, spiel in enumerate(spiele, start=1):
```

```
        print(f"\n--- Verarbeite {index}/{len(spiele)}: {spiel} ---")
        genres = suche_spiel_rawg(spiel)
        ergebnisse.append([spiel, genres])
        alle_genres.update(genres)
        time.sleep(1)  # kurze Pause um API-Limit einzuhalten

    # Genres alphabetisch sortieren für gleichbleibende Spalten
    genre_liste = sorted(alle_genres)

    # Ergebnis in neue CSV schreiben
    with open(ausgabe_datei, 'w', newline='', encoding='utf-8') as f:
        schreiber = csv.writer(f)
        kopfzeile = ["Spiel"] + genre_liste
        schreiber.writerow(kopfzeile)

        for spiel, genres in ergebnisse:
            zeile = [spiel]
            for g in genre_liste:
                zeile.append(g if g in genres else "")
            schreiber.writerow(zeile)

    print(f"\n[FERTIG] Daten gespeichert in '{ausgabe_datei}'")

if __name__ == "__main__":
    main()
```

Quellcode 3: Python Skript das mithilfe der eines RAWG-API Schlüssels die Datenbank von RAWG durchsucht. Nimmt eine CSV-Datei mit einer Liste an Spielen und gibt eine CSV-Datei mit einer Liste und zugeordneten Genres aus.

Anhang D: Doppelte Stichwort Suche

```python
import csv
CATEGORIES = {
    # Movement
    "Movement": [
    "Movement","Basics of Movement Algorithm",
       # Die Bibliothek der Stichwortliste wurde aus leserlichen Zwecken gekürzt.
    ]
}

seen = {}
for section, keywords in CATEGORIES.items():
    unique_keywords = set(k.lower() for k in keywords)  # deduplicate inside one section
    for key in unique_keywords:
        if key in seen:
            seen[key].append(section)
        else:
            seen[key] = [section]

# Prepare rows: one row per keyword, one column per section it appears in
rows = []
for kw_lower, sections_list in seen.items():
    if len(sections_list) > 1:  # Only include keywords that appear in multiple sections
        row = [kw_lower] + sections_list
        rows.append(row)

# Determine max number of columns needed
max_cols = max(len(row) for row in rows)

# Write to CSV
with open("Doppelte_Stichwoerter.csv", "w", newline="", encoding="utf-8") as f:
    writer = csv.writer(f)
    # Create header: Keyword, Section 1, Section 2, ...
    header = ["Stichwort"] + [f"Kategorie {i}" for i in range(1, max_cols)]
    writer.writerow(header)
    # Pad rows to match max columns
    for row in rows:
        writer.writerow(row + [""] * (max_cols - len(row)))

print("fertig")

```

Quellcode 4: Python Skript das doppelte Stichwörter zwischen den Kategorien der Bibliothek ermittelt. Es wird eine CSV-Datei mit den doppelten Stichwörter ermittelt

Anhang E: Gamedeveloper Stichwort Klammer Filter

```python
import csv
import re

# Input and output file names
INPUT_FILE = "Gamedeveloper_Stichwort_Ergebnisse.csv"
CLEAN_FILE = "Gamedeveloper_Stichwort_Klammerfrei.csv"
PAREN_FILE = "Gamedeveloper_Stichwort_mit_Klammern.csv"

# Regex patterns
s_suffix_pattern = re.compile(r"\(s\)$")
leading_empty_paren = re.compile(r"^\(\)")
paren_pattern = re.compile(r"\(.*?\)")

def process_entry(entry):
    entry = entry.strip()
    if not entry:
        return None, None  # Empty cell

    # Handle leading () case
    if leading_empty_paren.match(entry):
        entry = leading_empty_paren.sub("", entry).strip()

    # Check if ends with (s)
    if s_suffix_pattern.search(entry) and not paren_pattern.search(entry[:-3]):
        cleaned = s_suffix_pattern.sub("", entry).strip()
        return cleaned, "clean"

    # If there are any parentheses left, it's a paren case
    if paren_pattern.search(entry):
        return entry, "paren"

    return entry, "clean"

with open(INPUT_FILE, newline='', encoding='utf-8') as infile:
    reader = list(csv.reader(infile))

header = reader[0]
rows = reader[1:]

# Find indexes for Game and Website
game_index = header.index("Game")
website_index = header.index("Website")

# Prepare new CSVs
clean_rows = [header]
paren_rows = [header]

for row in rows:
    clean_row = []
    paren_row = []
    for i, cell in enumerate(row):
        if i in (game_index, website_index):
            # Always keep Game and Website intact in both files
            clean_row.append(cell)
            paren_row.append(cell)
            continue

        processed, category = process_entry(cell)
```

```
        if category == "clean":
            clean_row.append(processed if processed else "")
            paren_row.append("")
        elif category == "paren":
            paren_row.append(processed)
            clean_row.append("")
        else:
            clean_row.append("")
            paren_row.append("")
    clean_rows.append(clean_row)
    paren_rows.append(paren_row)

# Write outputs
with open(CLEAN_FILE, "w", newline='', encoding='utf-8') as outfile:
    writer = csv.writer(outfile)
    writer.writerows(clean_rows)

with open(PAREN_FILE, "w", newline='', encoding='utf-8') as outfile:
    writer = csv.writer(outfile)
    writer.writerows(paren_rows)

print(f" Created '{CLEAN_FILE}' and '{PAREN_FILE}' with Game and Website intact")

```

Quellcode 5: Python Skript das Stichwörter, die Teil eines anderen Wortes sind, werden ausgefiltert. (Mit Ausnahme von Stichwörtern nur mit „(s)“ als Endung). Stichwörter die Teil eines anderen Wortes sind wurden in dem Script aus Anhang B mit dem restlichen Teil des umschließenden Wortes in Klammern gesetzt formatiert. Dieses Script nimmt eine CSV-Datei mit den Ergebnissen der Stichwortsuche an und gibt zwei CSV-Dateien aus. Einmal mit Stichwörtern mit Klammern und einmal ohne.

Anhang F: CSV bereinigen und zusammenfassen

```
import csv
import os

# Whitelist-Kategorien
KATEGORIEN = {
    # Bewegung
    "Movement": [
        "Movement", "Steering", "Collision Avoidance", "Obstacle Avoidance", "Jumping",
       "Shooting", "Coordinated Movement", "Motor Control", "(enemy)Movement()", "(vehicle)Movement(stdboat)",
        "(unit)Movement()"

    ],
    # Pfadfindung
    "Pathfinging": [
        "Pathfinding", "Dijkstra", "A*", "Navigation Mesh", "Hierarchical Pathfinding",
        "Navmesh","(spatialquery)Navmesh(sample)", "(pimplrecast)Navmesh()"
    ],
    # Entscheidungsfindung
    "Decision Making": [
        "Decision Making", "Decision Tree", "Finite State Machine", "State Machine",
        "Hierarchical State Machine", "Behavior Tree", "Fuzzy Logic"," Goal-oriented Action Planning",
        "Tactical and Strategic AI"
    ],
    # Taktische und strategische KI
    "Tactical and Strategic AI": [
        "Tactical AI","Strategic AI","Influence Map","Convolution Filters","Cellular Automata"
    ],
    # Lernen
    "Learning": [
        "Intra-Behavior Learning",
        "Annealing","String Matching","ID3","Reinforcement Learning","Deep Learning",
        "generative AI", "Artifical Neural Network"
    ],
    # Prozedurale Content Generierung
    "Procedual Content Generation": [
        "Procedural Content Generation","Poisson Disk","Landscape Generation",
        "Height-Map","Maze Generation"
    ],
    # Brettspiele
    "Board Games": [
        "Game Theory", "Board Game"
    ]
}

# Whitelist flach in ein Set umwandeln und alles klein machen für case-insensitive Vergleich
WHITELIST = {item.lower() for sub in KATEGORIEN.values() for item in sub}

def bereinige_eintrag(eintrag: str) -> str:
    """Entfernt Klammern und deren Inhalt aus einem Eintrag."""
    import re
    return re.sub(r"\([^)]*\)", "", eintrag).strip()

def verarbeite_csv(eingabedatei):
    ausgabedatei = os.path.splitext(eingabedatei)[0] + "_bereinigt.csv"

    with open(eingabedatei, newline='', encoding="utf-8") as f:
        reader = list(csv.reader(f))

```

```
    kopfzeilen = reader[0]
    daten = reader[1:]

    # 1. Website-Spalte entfernen (Index 1)
    kopfzeilen.pop(1)
    for zeile in daten:
        if len(zeile) > 1:
            zeile.pop(1)

    # 2. Nur Spalten behalten, die in der Whitelist vorkommen (case-insensitive)
    behalten_indices = []
    for i, kopf in enumerate(kopfzeilen):
        if i == 0:  # immer die Spiel-Spalte behalten
            behalten_indices.append(i)
            continue
        spalten_eintraege = [zeile[i] for zeile in daten if i < len(zeile)]
        if any(eintrag.lower() in WHITELIST for eintrag in spalten_eintraege if eintrag):
            behalten_indices.append(i)

    kopfzeilen = [kopfzeilen[i] for i in behalten_indices]
    daten = [[zeile[i] for i in behalten_indices] for zeile in daten]

    # 3. Einträge bereinigen (Klammern entfernen)
    for r, zeile in enumerate(daten):
        for c, wert in enumerate(zeile):
            if wert:
                daten[r][c] = bereinige_eintrag(wert)

    # 4. Leere Spalten entfernen (außer Spiel)
    nicht_leer_indices = []
    for i, kopf in enumerate(kopfzeilen):
        if i == 0:  # immer Spiel behalten
            nicht_leer_indices.append(i)
            continue
        if any(zeile[i].strip() for zeile in daten):
            nicht_leer_indices.append(i)

    kopfzeilen = [kopfzeilen[i] for i in nicht_leer_indices]
    daten = [[zeile[i] for i in nicht_leer_indices] for zeile in daten]

    # 5. Leere Zeilen entfernen (außer Spiel-Spalte)
    bereinigte_zeilen = []
    for zeile in daten:
        if zeile[0].strip() and any(zelle.strip() for zelle in zeile[1:]):
            bereinigte_zeilen.append(zeile)

    # 6. Zeilen nach Spielname zusammenführen (erste Einträge pro Spalte gewinnen)
    kombiniert = {}
    for zeile in bereinigte_zeilen:
        spiel = zeile[0]
        if spiel not in kombiniert:
            kombiniert[spiel] = zeile
        else:
            for i in range(1, len(kopfzeilen)):
                if not kombiniert[spiel][i].strip() and zeile[i].strip():
                    kombiniert[spiel][i] = zeile[i]

    finale_zeilen = list(kombiniert.values())

    # Bereinigte CSV speichern
```

```
    with open(ausgabedatei, "w", newline='', encoding="utf-8") as f:
        writer = csv.writer(f)
        writer.writerow(kopfzeilen)
        writer.writerows(finale_zeilen)

    print(f"Bereinigte Datei gespeichert: {ausgabedatei}")

if __name__ == "__main__":
    EINGABEDATEI = "Gamedeveloper_Stichwort_Ergebnisse.csv"  # Eingabedatei hier setzen

    verarbeite_csv(EINGABEDATEI)


```

Quellcode 6: Python Skript, das eine CSV-Datei mit Stichwörtern bereinigt. Es entfernt die Website-Spalte, löscht leere Zeilen und Spalten (außer der Spiel-Spalte), bereinigt Einträge von Klammern, behält nur Spalten, deren Stichwörter in einer vordefinierten Whitelist vorkommen, und führt mehrere Einträge desselben Spiels zusammen, wobei der erste gefüllte Eintrag pro Spalte gewinnt. Alle Vergleiche erfolgen dabei unabhängig von Groß- oder Kleinschreibung. Das Skript erzeugt eine neue bereinigte CSV-Datei.

Anhang G: Gamedeveloper Oxford 5000 Filter

```python
import csv
import os

# File paths
oxford_csv_path = "oxford-5k.csv"
game_csv_path = "Gamedeveloper_Stichwort_klammerfrei_bereinigt.csv"

# Dynamically generate output filenames
base_name, ext = os.path.splitext(game_csv_path)
edited_csv_path = f"{base_name}_oxford5k{ext}"
removed_csv_path = f"{base_name}_entfernte_Stichwoerter{ext}"

# --- Helper: normalize word (basic lemmatization-like cleanup) ---
def normalize(word: str) -> str:
    word = word.lower().strip()

    # Handle plurals
    if word.endswith("ies") and len(word) > 4:
        return word[:-3] + "y"
    if word.endswith("es") and len(word) > 3:
        return word[:-2]
    if word.endswith("s") and len(word) > 2:
        return word[:-1]

    # Handle verb forms
    if word.endswith("ing") and len(word) > 4:
        return word[:-3]
    if word.endswith("ed") and len(word) > 3:
        return word[:-2]

    return word

# Load Oxford 5000 words into a set (normalized)
oxford_words = set()
with open(oxford_csv_path, newline='', encoding='utf-8') as f:
    reader = csv.DictReader(f)
    for row in reader:
        word = normalize(row['word'])
        if word:
            oxford_words.add(word)

# Read game CSV and process headers
with open(game_csv_path, newline='', encoding='utf-8') as f:
    reader = csv.reader(f)
    headers = next(reader)

    columns_to_keep = []
    columns_to_remove = []

    for i, header in enumerate(headers):
        header_lower = header.strip().lower()
        header_first_word = header_lower.split()[0]
        normalized = normalize(header_first_word)

        # Exception: always keep 'game' column
        if header_lower == 'game':
            columns_to_keep.append(i)
        elif normalized in oxford_words:
```

```
            columns_to_remove.append(header)
        else:
            columns_to_keep.append(i)

    # Read remaining rows with kept columns
    filtered_rows = []
    game_index = headers.index("Game") if "Game" in headers else 0
    for row in reader:
        new_row = [row[i] for i in columns_to_keep]
        # Keep only rows where at least one non-Game cell is non-empty
        if any(cell.strip() for j, cell in enumerate(new_row) if j != columns_to_keep.index(game_index)):
            filtered_rows.append(new_row)

# Write the edited CSV
with open(edited_csv_path, 'w', newline='', encoding='utf-8') as f:
    writer = csv.writer(f)
    writer.writerow([headers[i] for i in columns_to_keep])
    writer.writerows(filtered_rows)

# Write the removed columns CSV
with open(removed_csv_path, 'w', newline='', encoding='utf-8') as f:
    writer = csv.writer(f)
    writer.writerow(['Removed_Columns'])
    for col in columns_to_remove:
        writer.writerow([col])

print(f"Removed columns: {columns_to_remove}")
print(f"Edited CSV saved to: {edited_csv_path}")
print(f"Removed columns CSV saved to: {removed_csv_path}")

```

Quellcode 7: Python Skript, das eine bereinigte CSV-Datei mit Stichwörtern anhand der Oxford 5000 Wörterliste überprüft. Es normalisiert Stichwörter durch einfache Singularisierung und Verbform-Korrektur, entfernt Spalten, deren erster Wortteil in der Oxford-Liste vorkommt (mit Ausnahme der Game-Spalte), und erstellt zwei CSV-Dateien: eine mit den gefilterten Stichwörtern und eine mit den entfernten Spalten.

Anhang H: Anzahl der Genres aller nominierten Spiele auf den Game Awards

Anzahl	Genre																		
Jahre	Ac.	Ad.	RP	Sh.	In.	St.	Pl.	Si.	Ca.	Fi.	Pu.	MM	Sp.	Ar.	Ra.	Fa.	Cd.	Bo.	Ed.
2014	42	22	16	15	10	5	2	3	7	7	7	0	2	5	4	1	4	0	1
2015	44	22	21	14	15	2	9	7	4	4	5	2	6	4	4	0	1	0	0
2016	65	25	27	24	23	17	8	10	13	5	9	7	4	3	1	0	0	0	0
2017	88	44	37	35	27	15	16	7	11	5	5	7	8	9	6	1	0	0	0
2018	64	36	26	18	29	13	11	7	9	3	7	9	5	5	2	4	1	1	0
2019	81	47	31	38	27	9	6	7	8	6	12	9	6	1	3	5	1	0	0
2020	88	63	36	34	28	12	5	17	8	5	2	6	8	2	3	0	1	0	1
2021	93	62	35	27	21	13	16	7	2	5	2	7	4	6	5	1	0	0	0
2022	86	62	43	19	29	10	9	10	11	10	7	6	5	2	4	0	2	0	0
2023	77	59	31	12	19	19	2	9	8	7	5	3	3	4	6	0	0	0	0
2024	68	40	38	8	11	14	7	4	4	6	0	2	5	1	1	0	1	2	0
Ergebnis	796	482	341	244	239	129	91	88	85	63	61	58	56	42	39	12	11	3	2

Tabelle 1: zeigt die Anzahl der Genres aller nominierten Spiele bei den Game Awards im Zeitraum 2014–2024, aufgeschlüsselt nach Genre. Zur besseren Übersicht wurden die Genres in gekürzter Form dargestellt:
Ac = Action, Ad = Adventure, RP = Role-Playing Game, Sh = Shooter, In = Indie, St = Strategy, Pl = Platformer, Si = Simulation, Ca = Casual, Fi = Fighting, Pu = Puzzle, MM = Massively Multiplayer Online, Sp = Sports, Ar = Arcade, Ra = Racing, Fa = Family, Cd = Card, Bo = Board Games, Ed = Educational.
Die Spalte „Ergebnis“ weist jeweils auf die Summe der Genre hin.

Anhang I: Prozentuale Anzahl der Genres aller nominierten Spiele der Game Awards

Prozente	Genre										
Jahre	Ac.	Ad.	RP.	Sh.	In.	St.	Pl.	Si.	Ca.	Fi.	Pu.
2014	27,45%	14,38%	10,46%	9,80%	6,54%	3,27%	1,31%	1,96%	4,58%	4,58%	4,58%
2015	26,83%	13,41%	12,80%	8,54%	9,15%	1,22%	5,49%	4,27%	2,44%	2,44%	3,05%
2016	26,97%	10,37%	11,20%	9,96%	9,54%	7,05%	3,32%	4,15%	5,39%	2,07%	3,73%
2017	27,41%	13,71%	11,53%	10,90%	8,41%	4,67%	4,98%	2,18%	3,43%	1,56%	1,56%
2018	25,60%	14,40%	10,40%	7,20%	11,60%	5,20%	4,40%	2,80%	3,60%	1,20%	2,80%
2019	27,27%	15,82%	10,44%	12,79%	9,09%	3,03%	2,02%	2,36%	2,69%	2,02%	4,04%
2020	27,59%	19,75%	11,29%	10,66%	8,78%	3,76%	1,57%	5,33%	2,51%	1,57%	0,63%
2021	30,39%	20,26%	11,44%	8,82%	6,86%	4,25%	5,23%	2,29%	0,65%	1,63%	0,65%
2022	27,30%	19,68%	13,65%	6,03%	9,21%	3,17%	2,86%	3,17%	3,49%	3,17%	2,22%
2023	29,17%	22,35%	11,74%	4,55%	7,20%	7,20%	0,76%	3,41%	3,03%	2,65%	1,89%
2024	32,08%	18,87%	17,92%	3,77%	5,19%	6,60%	3,30%	1,89%	1,89%	2,83%	0,00%
Ergebnis	28,01%	16,96%	12,00%	8,59%	8,41%	4,54%	3,20%	3,10%	2,99%	2,22%	2,15%

Prozente	Genre							
Jahre	MM.	Sp.	Ar.	Ra.	Fa.	Cd.	Bo.	Ed.
2014	0,00%	1,31%	3,27%	2,61%	0,65%	2,61%	0,00%	0,65%
2015	1,22%	3,66%	2,44%	2,44%	0,00%	0,61%	0,00%	0,00%
2016	2,90%	1,66%	1,24%	0,41%	0,00%	0,00%	0,00%	0,00%
2017	2,18%	2,49%	2,80%	1,87%	0,31%	0,00%	0,00%	0,00%
2018	3,60%	2,00%	2,00%	0,80%	1,60%	0,40%	0,40%	0,00%
2019	3,03%	2,02%	0,34%	1,01%	1,68%	0,34%	0,00%	0,00%
2020	1,88%	2,51%	0,63%	0,94%	0,00%	0,31%	0,00%	0,31%
2021	2,29%	1,31%	1,96%	1,63%	0,33%	0,00%	0,00%	0,00%
2022	1,90%	1,59%	0,63%	1,27%	0,00%	0,63%	0,00%	0,00%
2023	1,14%	1,14%	1,52%	2,27%	0,00%	0,00%	0,00%	0,00%
2024	0,94%	2,36%	0,47%	0,47%	0,00%	0,47%	0,94%	0,00%
Ergebnis	2,04%	1,97%	1,48%	1,37%	0,42%	0,39%	0,11%	0,07%

Tabelle 2: zeigt die prozentuale Anzahl pro Jahr der Genres aller nominierten Spiele der Game Awards im Zeitraum 2014–2024, aufgeschlüsselt nach Genre. Zur besseren Übersicht wurden die Genres in gekürzter Form dargestellt: Ac = Action, Ad = Adventure, RP = Role-Playing Game, Sh = Shooter, In = Indie, St = Strategy, Pl = Platformer, Si = Simulation, Ca = Casual, Fi = Fighting, Pu = Puzzle, MM = Massively Multiplayer Online, Sp = Sports, Ar = Arcade, Ra = Racing, Fa = Family, Cd = Card, Bo = Board Games, Ed = Educational.
Die Spalte „Ergebnis“ weist jeweils auf den vollständigen prozentualen Anteil der Genre hin.

Anhang J: Anzahl der Spiele pro Kategorie, die mindestens einen Algorithmus haben

Kategorie	Anzahl der Kategorie pro Spiel
Movement	183
Decision Making	95
Pathfinding	89
Learning	61
Board Games	60
Procedual Content Generation	14
Tactical and Strategic	13
Gesamtergebnis	515

Tabelle 3: Anzahl der Kategorien pro Spiel, in denen mindestens ein valides Stichwort gefunden wurde

Anhang K: Anzahl der Spiele pro Kategorie, in denen mindestens ein Algorithmus gefunden wurde

Anzahl	Kategorie							
Jahr	Mov.	Dec.	Pat.	Lea.	BGS.	PCG.	TAI.	Summe
2014	18	12	11	6	4	3	3	19
2015	17	10	6	7	6	1	0	19
2016	23	12	7	9	3	2	3	24
2017	33	15	13	11	12	3	3	37
2018	24	9	9	9	9	2	2	24
2019	22	13	12	9	4	3	0	25
2020	19	8	9	12	4	2	2	21
2021	18	11	9	11	4	3	0	23
2022	23	13	12	12	5	2	0	29
2023	27	12	9	7	9	1	3	29
2024	10	4	5	8	4	1	0	15
Summe	183	95	89	61	60	14	13	207

Tabelle 4: Anzahl der Spiele pro Kategorie, in denen mindestens ein Algorithmus gefunden wurde.
Die Kategorien wurden für die Leserlichkeit abgekürzt: Movement = Mov., Decision Making = Dec.,
Pathfinding = Pat., Learning = Lea., Procedural Content Generation = PCG.,
Tactical and Strategic AI = TAI., Board Games = BGS.

Anhang L: Prozentuale Anzahl pro Jahr, der Spiele pro Kategorie, in denen mindestens ein Algorithmus gefunden wurde

prozentuale Anzahl	Kategorie						
Jahr	Mov.	Dec.	Pat.	Lea.	BGS.	PCG.	TAI.
2014	94,74%	63,16%	57,89%	31,58%	21,05%	15,79%	15,79%
2015	89,47%	52,63%	31,58%	36,84%	31,58%	5,26%	0,00%
2016	95,83%	50,00%	29,17%	37,50%	12,50%	8,33%	12,50%
2017	89,19%	40,54%	35,14%	29,73%	32,43%	8,11%	8,11%
2018	100,00%	37,50%	37,50%	37,50%	37,50%	8,33%	8,33%
2019	88,00%	52,00%	48,00%	36,00%	16,00%	12,00%	0,00%
2020	90,48%	38,10%	42,86%	57,14%	19,05%	9,52%	9,52%
2021	78,26%	47,83%	39,13%	47,83%	17,39%	13,04%	0,00%
2022	79,31%	44,83%	41,38%	41,38%	17,24%	6,90%	0,00%
2023	93,10%	41,38%	31,03%	24,14%	31,03%	3,45%	10,34%
2024	66,67%	26,67%	33,33%	53,33%	26,67%	6,67%	0,00%
Gesamtergebnis	88,41%	45,89%	43,00%	29,47%	28,99%	6,76%	6,28%

Tabelle 5: Prozentuale Anzahl pro Jahr, der Spiele pro Kategorie, in denen mindestens ein Algorithmus gefunden wurde. Die Kategorien wurden für die Leserlichkeit abgekürzt: Movement = Mov., Decision Making = Dec., Pathfinding = Pat., Learning = Lea., Procedural Content Generation = PCG., Tactical and Strategic AI = TAI., Board Games = BGS.

Anhang M: Erwartete Werte der Kontingenztafel zwischen Genre und AI-Kategorie

Erwarteter Wert	AI-Kategorie					
Genre	Movement	Decision Making	Pathfinding	Learning	Board Games	Gesamtergebnis
action	113,02	61,28	58,55	40,85	36,77	128
adventure	71,52	38,78	37,05	25,85	23,27	81
indie	45,03	24,41	23,33	16,28	14,65	51
rpg	44,15	23,94	22,87	15,96	14,36	50
shooter	37,97	20,59	19,67	13,72	12,35	43
strategy	22,96	12,45	11,89	8,30	7,47	26
casual	17,66	9,57	9,15	6,38	5,74	20
simulation	17,66	9,57	9,15	6,38	5,74	20
Gesamtergebnis	166	90	86	60	54	188

Tabelle 6: Tabelle mit den Erwartungswerten der Kontingenztafel zwischen Genre und AI-Kategorie

Anhang N: Kontingenztafel zwischen Genre und Stichwörter

Anzahl verschiedener Spiele	Stichwörter									
Genre	A*	BG	FSM	GAI	Ju	Mo	Pa	Sh	SM	Summe
action	27	37	17	25	53	90	47	58	27	381
adventure	19	22	11	18	43	55	32	38	23	261
casual	8	7	7	6	11	15	8	15	11	88
indie	8	16	5	12	30	32	16	21	13	153
rpg	8	15	5	6	24	31	16	25	10	140
shooter	11	13	8	10	21	31	25	25	13	157
simulation	5	7	6	5	9	13	7	5	8	65
Summe	86	117	59	82	191	267	151	187	105	1245

Tabelle 7: Kontingenztafel zwischen Genre und Algorithmen Stichwörter. Die Stichwörter wurden für die Leserlichkeit abgekürzt: Bord Game = BG, Finite State Machine = FSM, Generative AI = GAI, Jumping = Ju, Movement = Mo, Pathfinding = Pa, Shooting = Sh, State Machine = SM

Anhang O: Erwartete Werte der Kontingenztafel zwischen Genre und Stichwörter

Anzahl verschiedener Spiele	Stichwörter									
Genre	A*	BG	FSM	GAI	Ju	Mo	Pa	Sh	SM	Summe
action	26,32	35,80	18,06	25,09	58,45	81,71	46,21	57,23	32,13	381
adventure	18,03	24,53	12,37	17,19	40,04	55,97	31,66	39,20	22,01	261
casual	6,08	8,27	4,17	5,80	13,50	18,87	10,67	13,22	7,42	88
indie	10,57	14,38	7,25	10,08	23,47	32,81	18,56	22,98	12,90	153
rpg	9,67	13,16	6,63	9,22	21,48	30,02	16,98	21,03	11,81	140
shooter	10,84	14,75	7,44	10,34	24,09	33,67	19,04	23,58	13,24	157
simulation	4,49	6,11	3,08	4,28	9,97	13,94	7,88	9,76	5,48	65
Summe	86	117	59	82	191	267	151	187	105	1245

Tabelle 8: Erwartete Werte der Kontingenztafel zwischen Genre und Algorithmen Stichwörter. Die Stichwörter wurden für die Leserlichkeit abgekürzt: Bord Game = BG, Finite State Machine = FSM, Generative AI = GAI, Jumping = Ju, Movement = Mo, Pathfinding = Pa, Shooting = Sh, State Machine = SM

Anhang P: Transkriptionen Experteninterview 1 auf den GDD

Experteninterview 1

Interviewpartner: Experte 1
Datum: 09. Juli 2025 16:06 Uhr
Ort: German Dev Days 2025 im Saalbau Gallus in der Frankenallee 11 in Frankfurt am Main

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 09:16

I: Fallen dir irgendwelche weiteren Kategorien dazu gerade ein? Würdest du sagen das sind so alles an Algorithmen abgedeckt? Also für dich?

E: Für mich würde wahrscheinlich noch ein oh Gott, wie heißen die noch mal? Ein Commander AI glaube ich heißt das? Nicht ganz. Oder ein Scene Master AI fehlen? Das wurde unter anderem, lasse mich jetzt nicht Lügen, Ich glaube left for dead war das, wo das benutzt wurde. Und das immer schaut, dass die Tension über ein Level hinweg auf und abnimmt und niemals bestimmte Werte überschreitet. Indem es zusätzliche Gegner spawnt wenn die Tension low ist zu low oder zu low für den jetzigen Zeitpunkt. Also es ist in letzter Zeit nicht mehr so viel passiert, dann versucht es möglichst viele Gegner oder starke Gegner zu spawnen, um wieder ein bisschen Schwung reinzubringen. Und dasselbe es versucht den Leuten zu helfen, weil wenn die wenig Leben haben, wenn es ihnen schlecht geht, dann spawnt das zusätzliche Munition und Health-Packs speziell das mit Health-Packs kennt man ja auch aus anderen Spielen wie Resident Evil, die neuen Remastered haben das alle. Director-AI so heißt es.

I. Ich glaube vielleicht würde das eher so in Richtung Tactical AI fallen, wenn man das benden würde, aber das ist gut. Okay, dann die nächste Frage wäre:
Welche Algorithmen sind eurer Meinung nach? Einfach generell essentiell so. Wenn jetzt ein Student vorbeikommt und …

E: Essenziell zu lernen? Für Games im Allgemeinen? Für spezifische Games?

I: Für Games im Allgemeinen essentiell. Weil natürlich kann man sagen: Ok, welche Algorithmen man nutzen muss ist vom Spiel abhängig, aber so welche Algorithmen sind eigentlich so ziemlich sicher, dass man die nutzt und von denen man einfach zumindest wissen sollte wie die funktionieren, weil die einfach so häufig vorkommen. Eurer Ansicht nach

E: Also... Ich bin nur noch ganz- Movement und Pathfindigen, das fällt doch eigentlich so ziemlich zusammen, oder? Hätte ich jetzt behauptet

I: Ich habe die Kategorien tatsächlich aus so einem Buch entnommen und in Movement da fällt zum Beispiel noch so Steering Behavior und solche Sachen rein oder Physics AI, dass die zum Beispiel bei einem Sprung die Sachen predicted.

E: Okay, Also das ist dann relativ spezifisch tatsächlich. Also normalerweise Pathfinding und am besten ,würde ich behaupten, das was du hier beim min-maxing Board Games Drin hast, was kind of in Decision Making auch drin ist, wo das Spiel Entscheidungen je nach Gamestate entscheidet. Also weniger so Sachen, ganz spezifische Sachen wie Pathfinding. Wo gesagt wird okay das geht immer um Movement, dass irgendwas wohin muss, sowas, das kennt jeder. Das ist basic das ist gut erklärbar und deswegen wichtig, um allgemein Algorithmen zu lernen und zu lernen, sie zu nutzen, würde ich behaupten. Aber dadurch, dass, was du hier Bord Games min-maxing und ich würde da decision making dazuzählen, hast, ist, wo man lernt wie man versuchen kann, welche Arten von Möglichkeiten man hat dem Spiel zu sagen, dass es Entscheidungen anhand bestimmter Kriterien Fällen kann. Wenn man das einmal gelernt hat, kann man das in so ziemlich jedem Spiel wiederverwenden. Weil die Kriterien-. Es ist normalerweise Abstraktion zu Siegpunkten oder Interesse-Punkten und dann, die Suche nach Möglichkeiten des AIs oder des Spiels oder des NPCs die Siegpunkte oder interessenpunkten zu maximieren. Und diese Methode, wenn man die einmal gelernt hat und das ist ganz typisch bei Board Games zum Beispiel, aber auch Decision Making im Allgemeinen, kann man dann ab dem Zeitpunkt auf alles verwenden. Würde ich behaupten.

I: Hast du speziell irgendwelche Algorithmen dann schon bereits genutzt? Du kannst auch gerne ein Spiel nehmen, an dem du gearbeitet hast.

E: Also ich hatte mal ein Projekt explizit zur Director-AI in dem ich randomisiert- also der Spieler musste durch randomisierte Räume durchlaufen und das Director-AI hat versucht einen bestimmten Abfluss an Schwierigkeit aufrecht zu erhalten, egal wie gut oder wie schlecht der Spieler war. Das heißt es wurden mehr schwere Gegner gespawnt, mehr Health-Packs wurden gespawnt, wenn er schlechter war und die Räume waren weniger komplex aufgebaut, wenn er sich schwer getan hat und dann habe ich gesagt, okay das Ziel von diesem von diesen Tension Points, habe ich sie, glaube ich genannt, das Ziel davon nimmt über den Verlauf des Spiels mit an. Das heißt die Schwierigkeit wird immer. Höher, aber gleichzeitig wird von Anfang an-. Das können des Spielers fließt in die Bewertung dieser Schwierigkeit ein.

I: Da hast du meine letzte Frage schon ein bisschen so vorweggenommen und zwar: Wann und wo hast du die Algorithmen genutzt? Also speziell, wie wendest du die eben an? Vielleicht kannst du noch mal ein paar Sachen wiederholen so?

E: Zu welcher Frage jetzt noch mal? Zur Letzten?

I: Wann und wo hast du diese Algorithmen benutzt?

E: Okay, also wie gesagt in meinem Programmierprojekt damals. Indem ich spezifisch die Tension eines Spielers über sein Leben über seinen Fortschritt und über, wi viel Munition er hatte, sowie über die Anzahl an

Gegnern, die zu diesem Zeitpunkt auf dem Spielfeld waren. Habe ich quasi errechnet ein Wert genommen. Den musste der Algorithmus dann übersetzen in eine current Tension nenne ich es mal. Hat das verglichen mit maximale Tension bzw. der Tension die angesteuert wurde und je nachdem ob die technischen zu hoch oder zu niedrig war im Vergleich zu Zieltension habe ich in dem Spieler helfen lassen oder die Schwierigkeit angezogen.

I: Da ist ja dann dieser Punkt von Decision Making, also wie entscheidet der Algorithmus, ob er jetzt dem Spieler mehr HP Packs gibt oder mehr Enemies spawnt. Was ist so der Entscheidungsknackpunkt an der Stelle vielleicht gewesen?

E:Also du fragst jetzt in welcher Weise für was er sich entschieden hat? Ob er-

I: Nein, also was ihn dazu getrieben hat sich für was zu entscheiden? Wie hast du das gemacht?

E: Also basically ich habe ihm verschiedene Möglichkeiten gegeben. Es schwerer oder leichter zu machen. Und er hat sich in dem Fall ich glaube random entschieden, aber konnte sich nicht zu häufig wiederholen.

I Ah, okay.

E: Dass er nicht immer dieselbe Methode mehrmals versucht hat. Und ansonsten ging es hauptsächlich um Repetition. Wenn die Tension dann immer noch nicht nach zehn Sekunden oder so runtergegangen ist, hat das einfach noch mal versucht mit was anderem quasi okay.

I: Okay gut, das war's dann.

Anhang Q: Transkriptionen Experteninterview 2 auf den GDD

Experteninterview 2

Interviewpartner: Experte 2
Datum: 10. Juli 2025 12:16 Uhr
Ort: German Dev Days 2025 im Saalbau Gallus in der Frankenallee 11 in Frankfurt am Main

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 18:23

I: Du hast jetzt hier diese Kategorien gesehen. Fallen dir persönlich vielleicht noch irgendwelche Kategorien ein oder fehlt deiner Meinung noch irgendwas? Die sind sehr oberbegrifflich beschrieben, so ich weiß.

E: Aber also allgemein dafür Algorithmen?

I: Genau.

E: Also es gibt Sorting Algorithmen das wäre das erste und einfachste was mir einfallen würde. Um Listen zu ordnen, sind auch Algorithmen. Dann Validation, also zwischen Server und Client. Also für alles was zwischen Server und Client kommuniziert werden muss, weil das dann ja relevant ist, was der Server denkt in der Regel und weniger was der Client denkt.

I: Sind tatsächlich Punkte die der Millington in seinem Buch nicht erwähnt hat. Sorting Algorithmen.

E: Also das ist eigentlich so eins der basic Sachen, die mir als erstes einfallen würden. Das fällt mir jetzt erstmal spontan sein.

I: Das passt! Dann die nächste Frage: Welche Algorithmen sind deiner Meinung nach essenziell? Also einfach für die Videospielentwicklung. Wenn jemand anfangen würde, wo du sagst, okay, das musst du eigentlich wissen.

E: Ich meine wo wir gerade von Sorting gesprochen würden. Das ist schon eins der absoluten Basics. Das ist schon essentiell, weil man das für alles mögliche braucht. Also zum Beispiel ja, auch für Pathfinding ist ja die Basis auch noch mal Sorting weil beim Pathfinding willst du ja den kürzesten Weg aus einer Menge von Punkten finden und diese Menge musst du ja auch sortieren, damit du weißt welcher Weg kürzer ist als ein anderer. Ich würde sagen Pathfinding ist zwar einer der basic Algorithmen, aber du brauchst es nicht für jedes Game. Sondern du brauchst halt schon ein Game was auch Pathfinding braucht. Aber Es gibt sehr viele, die das brauchen. Also es ist auf jeden Fall auch eins der Basics. Essentiell... könnte ich sonst jetzt spontan, keine

Fachbegriffe nennen für Algorithmen. Ja, ist immer so. Was ich zumindest immer mache ist sehr speziell und wenn ich arbeite dann denke ich nicht drüber nach, was ist das jetzt für ein Algorithmus, sondern die einfachen Sachen hat man so verinnerlicht, da denkt man dann nicht mehr so drüber nach, auch so bestimmte Patterns. Und am Ende ist wie gesagt alles was Hinten rauskommt ein Algorithmus das muss dann kein allgemeiner sein, den man immer irgendwie wieder verwenden kann. Wie jetzt so ein A* Pathfinding kann man immer und überall benutzen weil man quasi immer vor das gleiche Problem gestellt wird und das immer gleich lösen kann. Aber das meiste andere sind ja Probleme, die wenn jetzt der speziell hat für das Spiel, was man hat. Und da brauchen wir da braucht man auch eine spezielle Lösung und speziellen Algorithmus.

I: Nur für den speziellen Fall?

E: Nur für den speziellen Fall, ja.

[...]

I: Die nächste Frage war, hast du irgendeiner diese Algorithmen schon mal genutzt?

E: Von den Sachen, die du jetzt hier stehen hast, das meiste wahrscheinlich eher im Studium, weil die meisten Sachen hier jetzt so AI related sind. Also Pathfinding, Decision Making das machen die AI Coder hauptsächlich, weil ich halt mehr so Gameplay mache für die Game Designer.
Procedual Content Generation habe ich auch im Studium gemacht. Jetzt gerade auch eher weniger. Das machen dann wahrscheinlich-. Also ich weiß nicht ob und was wir überhaupt hätten an Procedual Generation. Also ich weiß nicht, was wir genau machen was jetzt procedual ist und was jetzt handgemacht ist. Aber wenn dann wären es auf jeden Fall eher die Artist und Technical Artist und so, die dann Procedual Generation haben. Iso viele davon benutzen dann irgendeine Procedual Generation die sie kriege, aber die sie ja nicht selber schreiben und wenn man halt ein Technical Artist ist und man irgendeine spezielle Lösung wieder braucht, die man nicht von der Stange kriegt, dann macht man sich vielleicht eine eigene Procedual Generation. Und dann gibt es ja auch noch sowas wie irgendwie eine Map zu generieren. Einfach irgendwie eine Heightmap, um irgendein Terrain zu haben und das sind dann auch wieder eher die die Engine Leute die sich damit beschäftigen. Die Engine Coder. Da hätte ich dann jetzt auch wieder nichts mit zu tun. Also bei mir sind das dann immer eher spezielle Lösungen für spezielle Probleme.

I: Dann können wir eigentlich auch schon direkt zur letzten Frage kommen und zwar wann und wo hast du Algorithmen bereits genutzt?

[...]

E: Es würde helfen wenn ich mir noch mal die technische Definition von Algorithmen anschaue, um vielleicht ein bisschen besser zu differenzieren zu können.

[...]

E: Eine endliche eindeutige Handlungsvorschrift zur Lösung eines Problems.

I: Was ist das?

E: Das ist die Kurzdefinition von Algorithmus. Die endliche eindeutige Handlungsvorschrift zur Lösung eines Problems. Also die Anleitung zur Lösung eines Problems.

[...]

E: Also wenn wir drauf gucken, die Anleitung zur Lösung eines Problems. Also zum Beispiel ich sage wir haben das Problem Hit Validation. Also im Client Server Kontext der Client schießt auf irgendwas und der Client sagt ich habe das getroffen und das wir dann den Server gesendet und der Server muss sagen "ja das stimmt" oder "nee das stimmt nicht". Was man da dann auch noch berücksichtigen muss, ist halt, dass der Client die Welt anders sieht als der Server, weil es halt einen bestimmten Delay zwischen den beiden gibt und wenn du auf einen anderen Spieler schießt, dann sieht der die Welt ja noch mal anders als du, weil er ein anderer Client ist. Weil er kriegt den Schuss ja viel später ab, als du ihn geschossen hast, weil das ja erstmal zum Server muss und vom Server zu dem anderen Client. Da ist das Problem zu lösen, ob der Client wirklich getroffen hat oder nicht und zusätzlich auch das Problem ist sieht das für den Empfänger auch fair aus oder ist das irgendwie nach eins zwei Sekunden Delay und eigentlich, weiß ich nicht und in der Zwischenzeit hat man den anderen getötet und warum stirbt man dann noch? Die Problemstellung ist der Schuss. Quasi ist der Treffer valide? Und in der Lösung werden halt verschiedene Parameter ran gezogen, die auf dem Server schauen können, ob der Client in seiner Version der Welt recht haben könnte.

I: Gibt es dann auch solche Sachen wie von wegen das der Client schaut okay, der mit der besseren Verbindung oder sowas, dem traue ich eher, oder? Es gibt ja verschiedene Modelle, ob das dann alles nur im Client berechnet wird oder alles nur auf dem Server berechnet wird oder ob die sich die Arbeit teilen oder ob irgendein Priorisierungsalgorithmus wär das das wahrscheinlich.

E: Also in dem Fall ist quasi die Priorität auf dem Client, der sagt, dass er was getroffen hat. Weil in seiner Version der Welt hat er ja wahrscheinlich den anderen Spieler an der Position gesehen, wo er ihn auch getroffen hat, auch wenn auf der Server Version war der den er getroffen hat woanders. Aber wegen diesem Delay könnte es sein, dass der kleinen doch recht hat, obwohl der Server das anders sehen würde, wenn er zum gleichen Zeitpunkt geschossen hätte.

E: Das hat auch seine Grenzen. Also es gibt eine gewisse Toleranz und wenn man zu sehr der Server Version hinterher hängt dann wird man auch abgewiesen weil die Diskrepanz wird zu groß zwischen den verschiedenen Versionen sozusagen. Es werden verschiedene Parameter halt angeguckt. Zum Beispiel der Zeitpunkt. Also, dass der Client sagt ich habe zu dem Zeitpunkt geschossen und getroffen und das sagt er

dem Server und zu dem Zeitpunkt wo der Server sich das anguckt kann er halt schauen, wo war der Client in Wirklichkeit zu dem Zeitpunkt, zu dem er sagt dass er geschossen hat und wo war der den er getroffen hat zu dem Zeitpunkt und ist das plausibel, dass er von der Position wo er sagt, dass er geschossen hat geschossen hat, in die Richtung in die er dir sagt, dass er geschossen hat und ist es plausibel, dass er überhaupt noch Munition hatte. Ist dies ist das plausibel ist das plausibel. Und wenn der Servers sagt irgendwas stimmt da nicht. Das ist zu lange her oder du hättest aus dem Winkel gar nicht treffen können oder so, dann wird das halt abgewiesen.

[..]

I: Das heißt dann quasi: Der Algorithmus schaut was das Problem ist und ob das im Rahmen ist und wendet dann dementsprechend verschiedene Lösungen an.

E: Der [Algorithmus] kriegt ein Paket von remote Daten und muss die vergleichen mit seinen lokalen Daten und schauen ob die alle Sinn ergeben in dem Kontext.

I: Mache Spiele machen dass dann so, dass schon Client-Side Sachen schon entschieden werden.

[..]

E: Ja, slso es werden manche Sachen predictively gemacht, weil manche Sachen sehen komisch aus, wenn sie erst mit delay passieren. Also zum Beispiel die Munition ist ein gutes Beispiel, weil die Munition wird dir sofort abgezogen wenn du schießt auf dem Client und dann sagst du dem Server: Ich habe geschossen. Ist das okay? und dann sagt der Server entweder: ja okay du hast geschossen. Ich ziehe das jetzt auch ab und dann können alle so bleiben wie sie sind oder der Server sagt: nee das stimmt nicht. Du konntest nicht schießen aus irgendeinem Grund du warst noch beschäftigt oder so du hast gerade noch an der Leiter geklettert oder so. Der Server sagt nee du kannst nicht schießen. Hier ist so viel Munition wie du eigentlich hast und dann musst der Client das zurücksetzen.

I: Hättest du noch irgendwas zu Algorithmen hinzuzufügen?

E: Bei Algorithmen ist halt auch das Ding das das Problem was man versucht zu lösen kann super unterschiedliche Ausmaße haben. Also es kann so easy sein, wie ich habe eine Liste von Zahlen und will sie von größer nach kleiner sortieren. Dann hast du das Problem ungeordneter Haufen an Zahlen und ein Algorithmus der die Lösungs-Anordnung herstellt. Oder du kannst einfach super kompliziertes haben, wie Zum Beispiel, die Validierung ob ein Schuss wirklich getroffen hat. Deswegen ist es auch schwer zu sagen was sind so DIE Algorithmen? Weil es halt ne super krasse Bandbreite gibt, weil als Programmierer ist ja dein main Job Probleme zu lösen mit Hilfe von Algorithmen, also deswegen meine ich halt eigentlich alles was du machst sind halt Algorithmen, weil du musst halt die ganze Zeit irgendwelche Probleme lösen. Und that's it. Und man muss halt andere Tools noch nutzen um die Algorithmen zum laufen zu bringen, aber letztendlich macht man ja immer irgendein Algorithmus und manchmal muss man noch einen Überbau schaffen damit es überhaupt

funktioniert oder eine Datenbank sozusagen anlegen damit man die verwenden kann, während der Problemlösung. Aber ja, letztendlich läuft es halt darauf hinaus, das Game ist dann einfach nur eine Sammlung von von einer Million Algorithmen, die dann wiederum selber einen Algorithmus bilden alle zusammen.

Anhang R: Transkriptionen Experteninterview 3 auf den GDD

Experteninterview 3

Interviewpartner: Experte 3
Datum: 10. Juli 2025 15:11 Uhr
Ort: German Dev Days 2025 im Saalbau Gallus in der Frankenallee 11 in Frankfurt am Main

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 07:54

I: Ich habe hier einen Liste an Algorithmen oder eine Übersicht von Überbegriffen für Algorithmen aus einem Buch entnommen. Das heißt AI for Games und meine erste Frage wäre dahingehend: Jetzt wo du sie so siehst zu einem verstehst du einmal, was alles damit gemeint ist und fallen dir vielleicht noch weitere Kategorien dazu ein?

E: Also bei den meisten weiß ich was es ist. Ich wüsste jetzt bei Tactical and Strategic das ist sehr oberbegriffsmäßig ich wüsste jetzt nicht, was dazu genau gemeint wird. Ja den Rest kenne ich jetzt schon. Ja klar!

E: Ah fallen dir noch weitere Kategorien ein? Combat ist so ein Klassiker eigentlich wenn ich das natürlich wieder sehr bestimmtes games-spezifisch aber das ist ein Ding...

I: Was genau im Combat meinst du da?

E: Meinst über Algorithmik?

I: Ja

E: Unterschiedlichste Sachen, also ein Beispiel was wir bei uns im Spiel haben als 3d-Spiel. Ich weiß nicht ob es dann schon zu spezifisch für das Interview ist. Wir haben Waffen die einfach ganz normal in der 3D-Welt sind, aber wir haben eine isometrische Perspektive das heißt die Spieler sehen nur 2D. Wenn du einen Gegner schlägst, kann es mal passieren, ohne dass der Spieler sieht, dass er über etwas drüber schlägt und was wir dafür gemacht haben, ist es die Hitbox der Waffe halt nach unten zur erweitern auf dem Boden und noch ein

bisschen höher zu machen. Das du halt eine höhere Hitbox hast, die der Waffe aber folgt. Da war relativ viel Algorithmen drin.

I: Wieso? Also wo ist da der Algorithmen Part?

E: Ja da ist schon relativ viel Mathematik dahinter.

I: Wird die live größer die Hitbox?

E: Genau die ist echtzeitberechnet. Also die ist jetzt nicht einfach eine feste Hitbox weil, je nachdem wie du die hältst muss sie anders sein. Also wenn ich ein Schwert jetzt waagerecht halte ist es länger, als wenn ich es senkrecht halte. Und die passt sich halt daran an, dynamisch wie tief der Boden ist, wie weit du nach unten treffen kannst und wie weit sie nach oben geht, teilweise auch je nachdem wie hoch du hält, wie lange das Objekt ist. Das heißt das ist wirklich eine dynamische Berechnung der BoundingBox dieses Objektes, die dann rotiert wird zur Quaternion Identity das heißt in ihren Ursprungspunkt, um dann entsprechend verlängert zu werden und wieder zurückgesetzt zu werden. Das per Frame. Das ist relativ viel Rechenaufwand drin für was, was man im Endeffekt im Spiel gar nicht sieht.

I: Aber man spürt es halt?

E: Man spürt es weil du halt besser triffst. Weil sonst würdest du über Sachen schlagen. Ganz ganz oft.

I: Dann wäre die nächste Frage, welche Algorithmen sind so deiner Meinung nach essentiell, wenn es in Videospielprogrammierung geht? Also wenn ich jetzt ein Student bin welche sollte ich da schon safe mal wissen.

E: Also kennen ist immer gut. Algorithmen sind auf jeden Fall wichtig, oder natürlich super wichtig. Die Frage ist, du solltest sie kennen, du musst aber eigentlich die Theorie dahinter nicht verstehen, weil du sie einfach nur noch anwendest. Die Engines haben die alle schon drin. Wir arbeiten noch sehr viel mit Quaternionen, was man wahrscheinlich im mathe-Studium-. Also ich habe es nicht gelernt in meinem Studium überhaupt nicht, weil die viel zu komplex sind. Sind aber das A und O für Rotationen. Das bedeutet niemand macht meh Rotation mit Matrizen oder sowas, weil die halt Gimbal-Loch heavy sind. Alles wird mit Quaternionen gemacht, damit du die Gimbal-Locks verhindern kannst. Aber die Theorie dahinter verstehe ich selber auch nicht. Aber es ist was, was man oft benutzt und dann habe ich mich da wenigstens mal versucht einzulesen. Aber dreifach imaginäre zahlen oder sowas war schon sehr schwer. Es gibt sehr viel in diese Richtung was man benutzt, gerade ja Vektoren Sachen sind super wichtig auf jeden Fall. Translation, Rotation in die Richtung. Wie verlängere ich den Vektor? Normalising? Das sind so die Basiswerte, die in allen diesen Teilen drin sind, aber vor allen Dingen in Movement sind diese natürlich super wichtig.

I: Welche Algorithmen hast du dann schon bereits genutzt?

E: Also von diesen klassischen Algorithmen, die du meinst?

I: Dein Quaternius-Algorithmus hast du jetzt schon als Beispiel erwähnt.

[...]

E: Wir haben eine State Machine fürs Decision Making, die aber auch nicht von uns kommt. Wir haben das als Plugin, was auch ein Klassiker ist. Sehr viele der Sachen kannst du natürlich schon aus dem Store kaufen und in dem Fall haben wir das gekauft, weil wir ein gutes Plugin oder ganz früher schon mal gekauft für was anderes zum testen und benutzen das dann halt. Also das ist ganz oft die Realität im Spiel. Du benutzt die Algorithmen sehr viel, aber du hast sie nicht selber angewandt. In dem Fall halt auch. Also wir haben Decision Trees dafür. Ja mit relativ normalen Und- oder Oder-Verknotungen quasi. Das heißt Selectoren und Sequences mit denen du dann sehr schnell zusammenbauen kannst: Wenn das passiert mit der Condition tu das und so weiter.

[..]

E: Es ist ja auch ein Fakt. Ich kenne auch sehr viele Leute, die gar keine Assets mehr selber bauen, sondern alles kaufen. Und das klingt jetzt erstmal kacke. Aber wenn die das gut machen, wenn die auf den Stil achten und halt darauf achten, dass es wirklich gut passt, dann ist das nicht schlimm. Warum nicht? Wenn man eher so der Game Designer ist und ein geiles Spiel machen will, aber keine Art kann, dann kann man entweder jemanden da finden wenn man es hat. Geil. Wenn nicht, irgendwo kaufst es dir halt ne.

I: Ich weiß zum Beispiel für Dialoge. Zum Beispiel nutz ich ganz gerne immer Yarn-Spinner, weil das einfach kostenlos ist. Damit ich zum Beispiel kein eigenes Dialog-Tree-Tool ich machen muss.

E: Okay, das ist cool, ja.

I: Aber ich glaube ich schweife ein bisschen ab.

E: Ne alles gut.

[...]

I: Die letzte Frage wäre, wann und wo hast du dann so diese Algorithmen genutzt?

E: Quasi täglich. Klar wir gehen jetzt nicht so jetzt brauche ich den A* für den Fall und sowas, aber gerade, wenn es um diese base mathematischen Sachen geht, wie Detektoren, Quaternionen, das ist täglich Brot. Als Gameplay-Programmierer musst du immer irgendwas verschieben, Forces auf irgendwas anwenden mit Vektoren, um die Wurfreichweite von den Objekten zu bestimmen. Solche Dinge, also es gibt keinen Tag, wo ich keine Vektoren verwende.

I: Also würdest du sagen die sind überall verstecht so?

E: Also Ich habe vorher bei einem normalen IT-Dienstleister gearbeitet und da brauchte ich nie Vektoren, nie diese grundsätzlichen mathematischen Dinge. Seitdem ich im Gaming arbeite ist das täglich Brot wirklich jeden Tag. Was so state Machines angeht die Animationsnetzwerke sind alle State Machines. Die Decision Trees sind alle State Machines. Das kommt überall vor. Das Blueprint-System von Unreal ist ja quasi eine State Machine. Ein sich zusammengeklicktes Programmieren so also das sieht man überall. Das ist auch absolutes Basiswerk, das man immer braucht.

Anhang S: Transkriptionen Experteninterview 4 auf den GDD

Experteninterview 4

Interviewpartner: Experte 4
Datum: 10. Juli 2025 16:44 Uhr
Ort: German Dev Days 2025 im Saalbau Gallus in der Frankenallee 11 in Frankfurt am Main

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 07:18

[...]

I: [...] und fallen dir vielleicht noch weitere Kategorien so ein?

E: Ja also ich habe während meinem Studium auch viel KI programmiert und das ist so meine- Ich würde sagen so meine Leidenschaft in der Spielentwicklung, deswegen passt das überraschend gut.

[...]

E: Mir würde da jetzt nicht wirklich mehr einfallen. Also das ist eine ziemlich kompakte Liste an Themen in dem Bereich.

I: Die nächste Frage wäre. Welche Algorithmen sind so deiner Meinung nach essentiell für die Videospielentwicklung?

E: Im KI-Bereich?

I: Es muss nicht für eine KI sein. Das kann ja für alles möglich sein wie zum Beispiel Movement oder Pathfinding. Klar ist es natürlich immer vom Spiel abhängig welche man braucht, aber welche Algorithmen

sollte man schon einfach so kennen. Die kommen super oft vor und wenn du ein Spiel machst kannst du dir ziemlich sicher sein, dass dich irgendwann damit mal beschäftigen musst.

E: Ja, auf jeden Fall also. Ich habe eben schon mal Finite State Machine erwähnt. Diesen Algorithmus das kann man oftmals für KI verwenden, aber auch für andere Systeme, zum Beispiel als Game Manager, zum Beispiel, um zwischen verschiedenen Spiel-States zu wechseln. Beispielsweise habe ich das schon mal in einem Spiel für einen Screenshot Mode gesehen. Das Wechseln zwischen Screenshot Mode und normalem Gameplay oder auch Pausen Menü für sowas. Das heißt das findet man in vielen Orten, die auch nicht in KI sondern einfach in Spielen allgemein sind. Deswegen würde ich das auf jeden Fall als relevanten Algorithmus sehen in dem Bereich, also in Spielen. Ansonsten fällt mir eigentlich gar nicht so viel dazu.

I: Hast du irgendwelche Algorithmen bereits schon benutzt?

E: Ja also ich habe A* Pathfinding verwendet. Ebenfalls habe ich meinen eigenen Behaviour Tree zum Beispiel entwickelt. Prozeduale Generation von einer Welte habe ich ebenfalls schon entwickelt. Was so kein Algorithmus an sich ist, aber-. Die habe ich schon verwendet.

I: Dann die alles Entscheidende Frage. [...] Wann und wo hast du diese Algorithmen genutzt?

[...]

E: Alsoo was mir so einfällt ist zum Beispiel ein Inventarsystem. Das ist eine sehr rudimentäre Sache die in vielen Spielen vorkommt und da gibt es halt eine gute Vorgehensweise. Also einen ein Algorithmus, den man halt oftmals dann auch wiederfindet. Das habe ich in diesem Fall schonmal genutzt. Das wäre zum Beispiel auch verbunden mit einer Finite State Machine, dass man das aktiviert. Ansonsten habe ich halt ein Behaviour Tree, was auch außerhalb von Spielen oftmals verwendet wird, aber ebenfalls Spielen als KI Verhaltenslogik benutzt wird. Und das habe ich zum Beispiel bei der Implementation von Gegnern oder von Bossen verwendet.

I: Wofür hast du die Procedual Content Generation verwendet?

E:Das habe ich verwendet für die Generation von einer Landscape von der Welt quasi an sich. Wodurch quasi der Mesh also das Objekt von der Welt sich halt hoch oder runter und Berge geformt hat. Das habe ich da verwendet.

Anhang T: Transkriptionen Experteninterview 5 & 6 auf den GDD

Experteninterview 5 & 6

Interviewpartner: Experte 5, Experte 6
Datum: 10. Juli 2025 18:4 Uhr
Ort: German Dev Days 2025 im Saalbau Gallus in der Frankenallee 11 in Frankfurt am Main

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E5 (für Experte 5) und mit E6 (für Experte 6) abgekürzt.

00:00 - 07:21

[...]

I: Und die erste Frage wäre: Wenn ihr diese Liste seht – fallen euch noch andere Kategorien ein, die für Spiele relevant sind?

E5 & E6: Nein.

E5: Weil das deckt jetzt alles ab.

I: Also, die nächste Frage wäre: Welche Algorithmen sind eurer Meinung nach essentiell? Welche braucht man in Videospielen eigentlich fast immer?

E5: Movement auf jeden Fall. In jedem Spiel musst du dich irgendwie bewegen können – sei es über Steuerung mit Controller oder Maus und Tastatur. Vielleicht in einem reinen Puzzle-Game nicht, wo man wirklich nur puzzelt, aber sonst gehört das immer dazu.

E6: Alles, was irgendwie mit Physik zu tun hat, auch.

E5: kommt aufs Spiel an, auch Pathfinding oder Sowas kommt aufs Spiel an.

E6: Würde für dich auch sowas darunterfallen wie Health und Mana des Spielers zu checken und zu regenerieren?

I: Ja, auf jeden Fall. Aber generell, unabhängig von der Liste.

E6: Ich denke da diren an welchen Status hat der Gegner gerade – ist er aggressiv, hat er einen bestimmten Spieler im Visier? In MMO-Sprache: Auf wen hat er Aggro gepulled? Wie wird das getriggert – nur durch

Sichtkontakt oder auch durch Treffer von hinten? Das sind so typische Combat-Status-Sachen. Diese Zustandsabfrage.

E5: So ein Game Manager. Alles, was Variablen verwaltet, spielt da eine Rolle.

E6: Vielleicht dann auch dein Progress den du schon unlocked hast. Dein Spielerprofil.

I: Die nächste Frage. Welche Algorithmen habt ihr bereits genutzt?

E6: Ja, haben wir schon genutzt. Ich bin nicht so gut darin, das in Kategorien einzuordnen, aber ja – Game Manager, wie ich gerade aufgezählt habe. Natürlich haben wir auch Decision Making: Greife ich den Spieler an, oder nicht? Wenn der Sound weit genug weg ist, haben wir Regeln dafür. Wir lösen das bei uns über Behavior Trees. Das ist nicht unbedingt die beste Lösung, aber es hat sich bei uns so etabliert.

E5: Ein Navmesh und Pathfinding auf jeden FAll.

E5: Hinderniserkennung – ist da ein Objekt im Weg? Muss der Gegner drüberspringen? Sowas.

I: Okay. Dann die letzte Frage in diesem Zusammenhang: Wann und wo habt ihr diese Algorithmen genutzt?

E5: Gut, wir haben das schon etwas vorweggenommen. In Unreal sind viele Shooter-Sachen und Gegnerlogiken schon als Blueprints vorhanden. In Unity musst du das meistens selbst schreiben.

E6: Aber das Physiksystem – zum Beispiel Gravitationsberechnungen – ist auch bei Unity schon eingebaut. Also Rigid-Body-Simulationen und so etwas nutzen wir regelmäßig, auch wenn wir sie nicht selbst implementiert haben. Post-Processing ist ja im Endeffekt auch algorithmisch – Lichteffekte, Dynmic Lighting, Vignette. Bei uns färbt sich die Vignette je nach Gesundheitszustand anders.

E5: Zählt Menüsteuerung auch als Algorithmus?

I: Wenn ihr das algorithmis aufgebaut habt?

E5: Wenn man es runterbricht: ja. Man muss sehen, wechseln, Menüs bedienen – das ist im Prinzip auch ein Zustand.

E6: Genau. Wir haben zum Beispiel einen Game-Initializer, der die ganzen Manager-Klassen lädt und die Information hält: Ist das Spiel gerade pausiert, sind wir im Hauptmenü, oder sind wir in einer Szene – also im Level.

Anhang U: Transkriptionen Experteninterview 7 auf den GDD

Experteninterview 7

Interviewpartner: Experte 7
Datum: 10. Juli 2025 18:16 Uhr
Ort: German Dev Days 2025 im Saalbau Gallus in der Frankenallee 11 in Frankfurt am Main

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 07:01

I: Genau, also folgendes: Ich habe hier eine Liste an Algorithmen – das sind so Oberkategorien, die habe ich aus einem Buch entnommen. Das ist knapp 900 Seiten lang und heißt AI for Games. Genau. Und die erste Frage wäre: Fallen dir noch weitere Kategorien ein, die in dieser Liste fehlen könnten?

E: Hm, also wir hatten ja gerade schon über State Machines gesprochen.

I: Die packt er unter Decision Making.

E: OK. Ob mir noch weitere einfallen? Ich finde das Ganze teilweise sehr abstrakt. Zum Beispiel „Movement“

I: Das umfasst ja noch sehr viele Steering-Sachen.

E: Wo würdest du jetzt Physiks sehen auch in Movement?

I: Ja teils.

E: Aber es gibt ja auch Physics außerhalb von Movement. Also könnte man Game Physics als Kategorie eigentlich mit aufnehmen.

I: Würdest du das auch so sehen?

E: Ja. Ich würde Game Physics als Oberbegriff haben und Movement vielleicht drunter.
Was genau ist eigentlich mit „Learning AI“ gemeint?

I: Ah das ist eher generative KI, also neuronale Netze, die wirklich lernen.

E: Machine Learning?

I: Das muss ja nicht unbedingt Machine Learning sein. Live Machine Learning wäre für Spiele ein bisschen heavy – es gibt ja kaum Spiele, die direkt LLMs integriert haben.

E: Aber man hat eben einfache Systeme. Unser Spiel nutzt ja auch ein bisschen Sprachmodell-Integration. Wir haben ein Detektivspiel mit einer LLM integriert. Ich weiß nicht. Soll ich hier mein Spiel glazen?

I: Es geht mir hier nicht um konkrete Fälle, sondern eher um die generellen Kategorien. Also: Wenn dir nichts Weiteres einfällt, ist das auch okay. Die Liste deckt eigentlich ziemlich viel ab.
Dann direkt die nächste Frage: Welche Algorithmen sind deiner Meinung nach wirklich essentiell – also welche nutzt man unabhängig davon, um welches Spiel es geht, fast immer?

E: Also, was in fast jedem Spiel drin ist, sind Physics-Algorithmen. Und was wahrscheinlich in 100 % der Spiele vorkommt, ist eben Decision Making – wie du ja schon gesagt hast, meist über State Machines. Selbst sehr einfache Spiele nutzen mindestens State Machines, zum Beispiel um Dialoge abzuwickeln. Ich würde sagen, das sind die wichtigsten Algorithmen, die fast immer vorkommen.

I: Welche hast du davon selbst schon verwendet?

E: Physik natürlich nicht selbst programmiert, das ist in allen großen Engines integriert. Aber State Machines nutzen wir aktiv in unserem Spiel. Der Animator arbeitet darüber, und auch unsere Dialogstrukturen laufen über State Machines. Movement geht ja dann Hand in Hand mit Physics. Und wie schon gesagt wir haben bei uns auch ein Language Model integriert, auf das wir per API zugreifen. Das ist lokal in unserem Spiel eingebunden, aber technisch gesehen greifen wir auf ein externes Modell zu. Das ist ein Sonderfall, klar, aber unser Spiel ist da sehr speziell. Es gibt nichts Vergleichbares. Es gibt zwar Detektivspiele mit LLMs für Dialoge, aber unser Ansatz ist nochmal anders.

I: Verstanden. Aber um es klar zu haben: Wenn es um essenzielle Algorithmen geht, würdest du eher Physik und State Machines nennen.

E: Genau.

I: Dann die letzte Frage: Wann und wo hast du diese Algorithmen eingesetzt?

E: Ja, das habe ich teilweise schon vorweggenommen. Also, State Machines sind bei uns extrem wichtig für Dialoge und Animationen. Physics für Interaktionen mit 3D-Objekten – also Gravitation, Kollisionen, das Übliche. Prozedurale Generierung nutzen wir in unserem Spiel nicht, auch wenn das natürlich in anderen Spielen viel vorkommt. Und es gibt auch Verfahren wie Test-Algorithmen mit Zielfunktionen, die dann gelöst werden – sowas kennt man auch. Aber im Kern bleiben es für mich die beiden Hauptpunkte: Physics für die 3D-Welt und State Machines für Dialoge und Animationen

Anhang V : Transkriptionen Experteninterview 8 auf der Gamescom

Experteninterview 8

Interviewpartner: Experte 8
Datum: 21. August 2025 19:13 Uhr
Ort: Gamescom in der Indie Arena Booth, Messeplatz 1 Köln

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 6:46

I: Okay. After looking at this list, could you think of any additional categories, perhaps specifically for algorithms or AI? Take your time.

E: I think the categories are broad enough to basically cover everything. Especially areas like decision-making are so wide that you can fit almost anything under them. In game development, there are many situations where you come up with an algorithm that is not necessarily profound. Often you just have a very practical problem that needs to be solved quickly and a simple algorithm will do. These solutions can become messy and non confirmative. I think such cases sometimes cannot be classified neatly. A mathematics professor might not recognize anything in them because it looks more like a "mess". But that’s part of game development.

I: Could you give an example? with puzzles or sorting lists?

E: Yes, I’ve written algorithms for UI elements to sort them correctly. Actually is "sorting" on your list? I don’t see "sorting" Maybe it is within one of them but sorting algorithms are an important category.

I: Good. Then let’s move on to the next question, which may be even more important: Which algorithms do you consider the most essential in game development? Perhaps the top three?

E: That’s a very difficult question. As a programmer, my craft is solving problems, and algorithms are just different tools for that. But I can say which ones are particularly common. Pathfinding is extremely widespread as you mentioned earlier before the interview. Many engines provide ready-made solutions, but those often don’t suffice, so you end up having to implement your own. Pathfinding is therefore at the top.

Secondly, I would name sorting. Sorting algorithms are extremely common, usually solved with simple methods, but they are everywhere. And third here I am a bit biased I would say procedural generation. This depends heavily on the project, but the field is vast and complex, which makes it essential as well.

I: So pathfinding, sorting, and procedural generation?

E: Exactly. The first two are like the "bread and butter" of programming, while the third one depends on the project but is a wide and important field.

I: Do you see any of these categories as particularly important for the future?

E: In fact, they are all important for the future. Algorithms don’t radically change over time. Of course, we develop better ideas and solutions, but every game needs algorithms. It’s hard to say which one has a "brighter future" when they are all indispensable.

Of course, generative AI is an exciting new topic that will change a lot. But for programmers who write custom algorithms for specific problems, the answer remains: they are all important and all will continue to be used.

I: Understood. Let’s move on to the next question: Which algorithms have you personally used?

E: Many, really many. I’ll limit myself here to my latest project developing bots for a game For that, I used a lot of pathfinding algorithms, for example for handling tools or gadgets, but also for general navigation.

Another major aspect was decision-making. When you build a bot, almost everything comes down to the question: What should it do, when should it do it, and how should it do it? For all of that, you need decision-making algorithms. Pathfinding answers the "How do I get there?" while decision-making answers the "What do I do?"

I: Where and how have you used algorithms in practice?

E: In recent months, mainly in bot development. The challenge was to create bots for a competitive game that was not originally designed for bots. The bots therefore had to act on a level comparable to human players which is a very complex task.

Besides that, I have also worked on procedural projects that I enjoyed a lot. A highlight for me was the so called "Wave Function Collapse" algorithm. With it, you can, for example, fill a grid in a Sudoku-like fashion, very exciting.

I: That sounds demanding. Do you work mostly "hard-coding" these solutions?

E: Yes, very much so. Especially with algorithms needed for specific projects, there are hardly any generic solutions. Many algorithms you write are extremely specific and will never be reused, because they were made only for that exact problem. Only a few, like the "Wave Function Collapse" algorithm, can be applied more generically.

I: Thank you very much, that was very insightful. Would you like to add anything at the end?

E: Just that algorithms are great. I truly enjoy developing them, and I hope many programmers will continue to share this enthusiasm in the future. They are such a central part of our work.

Anhang W Transkriptionen Experteninterview 9 auf der Gamescom

Experteninterview 9

Interviewpartner: Experte 9
Datum: 21. August 2025 19:34 Uhr
Ort: Gamescom in der Indie Arena Booth, Messeplatz 1 Köln

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 13:43

I: Also die erste Frage wäre. Fallen dir vielleicht noch weitere Kategorien ein?

E: Nicht spontan. Ich wette, das man alles was mir an Algorithmen einfällt bestimmt irgendwo reinkriegen kann und wenn man keine Kategorie findet, dann packt man es halt in Board Games. Vielleich wenn ich weiter drüber nachdenken würde, würde es mir einfallen.

I: Kommen wir eigentlich zu aller wichtigsten Frage und zwar welche Algorithmen sind so deiner Meinung nach essentiell?

E: Kommt aufs Spiel an und selbst wenn du ein sehr konkretes Spiele Genre hast, wie RTS kannst du sagen ja ich brauch auf jeden Fall Pathfinding? Ja nee, du kannst auch sagen ich kann ein RTS machen, in dem du kein Pathfinding brauchst oder wo eine ganz andere Art von Pathfinding ganz andere Algorithmen benutzt. Oder unter den Pathfinding-Algorithmen eine andere Wahl triffst, als bei anderen Spielen.

I: Ich stell die Frage auch gerne um, was würdest du Studenten und Anfängern da empfehlen?.

E: Also ich bin ich bin Mathematiker. Ich habe Mathematik studiert. Ich habe Informatik studiert. Ich habe Mikroelektronik studiert. Alles was man damals noch im Diplom an Algorithmen kennengelernt hat ist gut.
Alles was du an Algorithmen kennenlernst ist gut, weil es eine Basisgrundlage ist. Wenn du ein Spiel entwickelst, sollst du nicht überlegen "Okay, welchen Algorithmus brauche ich", dass ist der falsche Ansatz. Du solltest überlegen was möchte ich erreichen oder welches Problem habe ich? Und dann schaust du das Problem genau an und guckst okay, was habe ich für eine Ausgangssituation? Was soll die Endsituation sein? Wie komme ich dahin? Und dann kannst du dich entlanghangen all dem, was du schon weißt. Wenn du einen Algorithmus kennst nächster Algorithmus ist ein schöner Start, meinetwegen für Movement und denkt hey,

das ist genau das Richtige. Aber das muss es nicht sein. Du musst in dem Moment anfangen zu gucken, was gibt es in dem Bereich noch, was gibt es für andere Sachen?

Ah! Kategorie: Sortier Algorithmen. Schönes Beispiel: es gibt super viele effektive Sortieralgorithmen du hast von Quicksort, Mergesort ganz ganz viele. Natürlich kennt jeder auf den guten alten Bubblesort und jeder lernt im Studium, dass man niemals Bubblesort benutzen sollte.

Wir haben ein Spiel gemacht, ein MMO mit ganz viel Spielern und wir hatten Probleme unsere Spieler Statistiken, die nach jedem Spiel aktualisiert wurden, für Millionen von Spielern, in Echtzeit schnell genug zu aktualisieren. Die Solution war Bubblesort. Weil die Voraussetzung war nicht "Oh wir haben hier ganz viele Daten. Wir müssen die sortieren.", sondern "Nein. Wir haben die ganz viele sortierte Daten". Wenn sich etwas ändert von den Stats und dann sind sie nicht mehr richtig, aber wir wissen dass die Stats immer dicht dran sein werden weilsie niemals große Sprünge machen. Von Platz 200 auf 210 oder von 190, aber nicht riesige Sprünge und dem Moment wenn du rüber bubbelst kannst du sie einfach nur lokal ein bisschen bubbeln und du kannst parallel dazu neue Datensätze mit einbubbeln. Wenn du neue Spieler hast, die das erste Mal da sind, das heißt du iterierst drüber, du babbelst hoch und du nimmst einfach die Datensätze, die neu sind mit. Und während du durch bubble-sortiert du sie ein und sortiert alles andere mit. Es ist nicht präzise. Aber es reicht, dass du mehrere Aufgaben gleichzeitig machen kannst in einen Durchlauf. Keiner ist im erst Anlauf perfekt, aber das reicht, weil die Daten sich nun in echtzeit ändern.

Und das ist von meinem Ansatz, der sich mit Algorithmen beschäftigt, er Lösungen findet immer mehr dieses: Alles was du weißt, ist eine Grundlage sind Startpunkt, von wo aus du gute Lösungen finden kannst. Kategorien kennen ist gut, wenn man sich in Literatur auskennt, aber immer erstmal suchen und dann nicht eine Standardlösung zu nehmen, sondern was brauchst du für dein Problem?

I: Die nächste Frage wäre, welche Algorithmen hast du bereits genutzt?

E: Zu viele, zu viele. Ich habe mein erstes Spiel 1999 gemacht. Ich mache jetzt für Industriespiele, lass mich viel bezahlen, seit 18 Jahren spiele. Ich arbeite hauptsächlich in Bereichen wo Algorithmen gebraucht werden. Ich bin in AI, Physik, also aus all den Kategorien die hier aufgelistet sind. Also ich habe in Movement ganz viel gemacht. Ganz viele Eigenentwicklungen auch aber basierend auf Sachen die ich gekennzeichnet habe.

Also ich finde Movement ist so allgemein. Du kannst so viele schöne Sachen machen.
Ich habe Pathfinding genutzt. Ich habe Decision Making genutzt. Ich habe Taktische und Strategische Sachen gemacht. Ich habe MinMax AI für Board Games, für meine eigenen Brettspiele eigene Ais geschrieben.

Lernende KI habe ich benutzt bei meinem allerersten Job bevor ich gelernt habe, dass AI und Spiele-AI zwei unterschiedliche Dinge sind. Du möchtest nicht eine AI machen, die sich intelligent verhält oder effektiv ist. Du möchtest eine AI machen, die Spaß bringt und Das Gamedesign stützt.Dann kommts aufs Spiel drauf an.Da musst du deine eigenen Sachen wieder erfinden.

Procedual Content generation habe ich wahrscheinlich von allem am wenigsten gebraucht. Weil das schwierigste ist aussagekräftige Momente oder etwas für den Spieler zu machen. Das ist super schwierig mit prozedualen zufällig generiertem Content zu schaffen.

Genauso mit der AI. Du möchtest, dass der Spieler sie lesen kann. Dafür muss sie klar lesbare Entscheidung treffen und nicht irgend nen Wischi-Waschi.

I: Okay dann die letzte Frage wäre: Wann und wo hast du diese Algorithmen benutzt? Wie sah die Implementierung aus?

[...]

E: Ein konkretes Beispiel: Ich bin in eine Firma gekommen, die haben gerade ein Moba gemacht mit großen Schlachtraumschiffen. Stell dir vor nicht so ein kleiner Fighter die sich schnell bewegt, sondern richtig großes Raumschiff und sie hatten schon versucht eine Physik dafür zu haben, wie sich das verhalten soll und haben das immer wieder weggeworfen haben es nicht geschafft.

Als ich in die Firma kam, war ich der nächste Programmierer, der es ausprobieren durfte. Das Erste, was ich gemacht habe, war, mir den verantwortlichen Game Designer zu schnappen und mit ihm in einen Raum zu gehen. Ich sagte: „Mach bitte mal die Augen zu. Stell dir das Spiel vor, in dem die Physik eine Rolle spielt, und erzähl mir, was du siehst, was du hörst und wie sich das anfühlt."
Er beschrieb ein großes Schlachtschiff, das zunächst stillsteht. Dann muss es erst die Maschinen hochfahren. Man spürt, wie die Motoren arbeiten, wie alles rüttelt, man hört das Vibrieren, und das Schiff nimmt langsam Fahrt auf, bis es wirklich schnell ist. Wenn ein so großer Kreuzer eine Kurve fährt, dauert es sehr lange, bis sich überhaupt etwas verändert. Auch beim Abbremsen braucht es Zeit.
Er beschrieb außerdem die Situation, wenn zwei große Schlachtkreuzer zusammenstoßen: Man hört, wie das Metall quietscht, wie ein Ruck durch das ganze Schiff geht, wie sie zum Stehen kommen und wie schwierig es ist, wieder herauszukommen, wenn sie versuchen, den Rückwärtsgang einzulegen. Das Metall knirscht, verhakt sich, bis es irgendwann nachgibt, ein Ruck durchgeht und sich die Schiffe wieder voneinander lösen.

Das habe ich als Input genommen, um mir ein physikalisch-mathematisches Modell zu überlegen, das ich implementieren könnte. Nachdem wir auseinandergegangen waren, habe ich in den folgenden Tagen an einem Modell gearbeitet. Ich habe ein wenig in der Physik recherchiert und vorgeschlagen: „Was hältst du davon, wenn wir die Raumschiffe so behandeln, als würden sie sich in einer Flüssigkeit bewegen?"
Der Weltraum wäre in diesem Modell also wie eine Flüssigkeit. Das bedeutet, dass man zunächst Kraft aufbringen muss, um das Schiff in Bewegung zu setzen, und dass es eine Art Maximalgeschwindigkeit gibt. Niemand möchte echte Physik im Spiel, sondern eine Physik, die das Gameplay unterstützt. Wir haben dann viele Szenarien durchgespielt: Was passiert, wenn dies oder jenes eintritt?

Schließlich kamen wir zu dem Punkt: „Wie ist es, wenn zwei Schiffe zusammenstoßen?" Wir beschlossen, die Schiffe nicht als starre Körper zu behandeln, sondern so, als bestünden sie ebenfalls aus Flüssigkeit, aber

einer sehr zähen Flüssigkeit. Das bedeutet: Wenn sich die Schiffe ineinander schieben, verändert sich das Medium. Zuvor bewegte man sich schnell durch eine leichte Flüssigkeit, doch sobald man auf ein anderes Raumschiff trifft, wird das Medium zäh. In diesem Moment reicht die eingebrachte Kraft nicht mehr aus, und das Schiff bleibt hängen. Man kann sich nicht sofort herausdrehen und muss gegen diese Masse ankämpfen, bis die Überlappung vorbei ist. Erst dann bewegt sich das Schiff wieder frei.

Diese Mechanik konnten wir nutzen, um auch Soundeffekte zu erzeugen. Wir hatten die Trigger, wir wussten, wann sich das Medium änderte, und konnten diese Informationen als Input für einen Soundalgorithmus verwenden. Dann schrieb ich einen Prototypen: Wegwerf-Code in Unity, mit einer kleinen Szene und zwei rechteckigen Objekten, die sich wie in einer Flüssigkeit bewegten. Einer war mit WASD steuerbar, der andere mit den Pfeiltasten. Das Ergebnis war: Trägheit, langsame Bewegung, Kurven waren schwer, und wenn die Objekte ineinander stießen, war es mühsam, sie wieder auseinanderzuziehen. Zwei Daumen hoch. Es funktionierte.

Danach habe ich zwei Wochen lang durchgehend programmiert, bis das System implementiert war. Dabei stellten wir noch etwas fest: Wenn Asteroiden im Spiel auftauchten, ergab sich ein Problem. Bei echter Physik würde ein Schiff nach einer Kollision mit einem starren Objekt in Rotation versetzt werden. Das führte jedoch dazu, dass es sich immer weiter in das Hindernis hineinverkantete. Fürs Gameplay war das katastrophal, weil der Spieler steckenblieb und nicht weiterspielen konnte.
Unsere Lösung: Wir haben das Vorzeichen der Newtonschen-Physik einfach invertiert. Wenn ein Schiff mit etwas kollidierte, wurde es nicht in das Hindernis hineingedreht, sondern herausgedreht. So konnte sich der Spieler schnell wieder befreien und weiterspielen.

I: Ok das war interessant. Hast du vielleicht noch irgendwelche Sachen hinzuzufügen?

E: Nach 18 Jahren in der Branche sammelt man eine Menge. Aber meine Empfehlung ist: Immer eigene Gedanken machen. Alles, was man zuvor gelernt hat, hilft, wenn man mehr Ansatzpunkte hat. Wenn man nur einen einzigen Algorithmus kennt, kann man nur lokal optimieren. Kennt man mehrere, kann man den gesamten Raum zwischen den Algorithmen erkunden und sehen, was in der eigenen Situation am besten passt.
Viel viel anschauen, viel viel lernen, viel experimentieren und eigene Gedanken machen!

Anhang X: Transkriptionen Experteninterview 10 auf der Gamescom

Experteninterview 10

Interviewpartner: Experte 10
Datum: 21. August 2025 20:09 Uhr
Ort: Gamescom in der Indie Arena Booth, Messeplatz 1 Köln

Im Transkript wird der Interviewer Jonathan Buss mit I (für Interviewer) und der Interviewpartner mit E (für Experte) abgekürzt.

00:00 - 6:56

I: Die erste Frage ist: Wenn du diese Liste hier siehst, fallen dir noch weitere Kategorien ein?

E: Auf Anhieb nicht. Die Kategorien sind sehr breit gefasst, deshalb finde ich es schwierig, weitere hinzuzufügen.

I: Dann kommen wir zur nächsten Frage: Welche Algorithmen sind deiner Meinung nach essentiell?

E: Sprechen wir nur über die Kategorien oder über bestimmte Algorithmen allgemein? Also, wenn ein Anfänger zu mir kommt und fragt: „Welche Algorithmen sollte ich kennen?“, dann würde ich sagen: Einige grundlegende Algorithmen sind essenziell, besonders wenn man sie auf verschiedene Datenstrukturen anwenden kann.
Es macht Sinn, zu wissen, wie sie funktionieren. Zum Beispiel, wenn man eine Tilemap hat darauf A* anwenden kann oder in der Geometrie arbeitet, ist es hilfreich, passende Algorithmen anwenden zu können. Auch ist es wichtig, sich der unterschiedlichen Arten bewusst zu sein, damit man für das jeweilige Game die richtigen auswählen kann.

Für die Games, die ich mache, ist prozedurale Generation sehr relevant. Aber das hängt stark vom Genre ab: Manche Spiele haben gar keine prozedurale oder procedural Content Generation, bei anderen ist das die gesamte Identität. Gleiches gilt für NPCs: Manche Games haben keine, andere bestehen fast nur daraus.

Deshalb ist es schwierig, allgemeingültig zu sagen, welche Algorithmen essentiell sind. Für meine Projekte sind prozedurale Generation und Decision Making besonders wichtig, also Dinge wie Behavior Trees oder State Machines. Oft werden auch Algorithmen genutzt, die Heuristiken verwenden, um etwas zu evaluieren und dem Computer die Richtung vorzugeben oder dem Spieler passenden Content zu liefern, je nachdem, was er gerade tut.

I: Welche Algorithmen hast du bereits konkret genutzt?

E: Wie gesagt, eigentlich schon verschiedene. Machine Learning ist bei mir eher selten. Meist reicht gutes Decision Making. Man braucht keinen Computergegner, der tatsächlich lernt, wenn er schon auf Basis guter Regeln stark genug ist. Einmal habe ich ein Racing Game gemacht, bei dem wir einen Learning-Algorithmus ausprobiert haben, um die beste Racing Line zu finden. Aber in kommerziellen Games habe ich so etwas nie eingesetzt. Das war nie nötig und auch oft zu buggy und unvorhersehbar.

Minimax oder klassische Analyse-Algorithmen habe ich nie in einem kommerziellen Game verwendet, aber natürlich schon aus Eigeninteresse implementiert. Zum Beispiel für Projekte an der Uni oder privat bei Spielen wie Dame. Das kann interessant sein, aber in Games wirkt es oft zu mechanisch. Für Spieler ist es meistens schöner, wenn sie vorhersagen können, wie der Computer sich verhält, und sich daran anpassen können. Gerade in Strategiespielen ist das sehr wichtig.

I: Wann und wo hast du solche Algorithmen schon angewendet?

E: Zum Beispiel beim Movement von Truppen auf einem Tile Grid, also beim Finden von kürzesten Pfaden oder allen Pfaden. Dann mit einem Decision-Making-Algorithmus heuristisch abwägen, welcher Pfad genutzt werden sollte, um das Spielerlebnis nicht unfair zu machen. In unserem aktuellen Game ist es zum Beispiel so, dass wir nicht wollen, dass der Gegner immer den absolut besten Pfad nimmt, wenn der Spieler dort gar keine Verteidigung hat. Das würde sich unfair anfühlen. Stattdessen wählen wir bewusst Pfade, die für den Spieler eine Herausforderung darstellen, aber nicht sofort zur Niederlage führen.

Generell denke ich, dass es sinnvoll ist, einige grundlegende Algorithmen zu kennen. Klassiker wie A*-Search sind essentiell, wenn man mit Nodes oder Tiles arbeitet. Vieles ergibt sich aber organisch mit der Zeit. Man muss nicht von Anfang an eine Sammlung von Algorithmen auswendig können – wichtiger ist, die Fähigkeit zu haben, sich diese bei Bedarf anzueignen.

Anhang Y: Explizit als essenziell genannte Algorithmen aus den Experteninterviews

Experte	Movement	Pathfinding	Decision Making	Tactical/ Strategic	Learning	Procedural Generation	Board Games	Sorting	Physics
Experte 1	0	1	1	0	0	0	0	0	0
Experte 2	0	1	0	0	0	0	0	1	0
Experte 3	1	0	1	0	0	0	0	0	1
Experte 4	0	1	1	0	0	0	0	0	0
Experte 5	1	0	0	0	0	0	0	0	0
Experte 6	0	0	1	0	0	0	0	0	1
Experte 7	0	0	1	0	0	0	0	0	1
Experte 8	0	1	0	0	0	1	0	1	0
Experte 9	0	0	0	0	0	0	0	1	0
Experte 10	0	1	1	0	0	1	0	0	0
Summe	**2**	**5**	**6**	**0**	**0**	**2**	**0**	**3**	**3**

Tabelle 9: Explizit erwähnte Algorithmen oder Kategorien wurden auf die Frage ob es essenzielle Algorithmen gäbe aus allen Experteninterviews gezählt und in dieser Tabelle zusammengefasst.

Tabelle 19: Zeigt alle Erwähnungen bezüglich Algorithmen aus allen Experteninterviews

Anhang Z: Algorithmen Erwähnungen der Experteninterviews

Experte	Movement	Pathfinding	Decision Making	Tactical/ Strategic	Learning
Experte 1	Ja (Movement & Pathfinding zusammen)	Ja	Ja (Decision Making, Director AI)	Ja (Director AI)	Nein
Experte 2	Nein	Ja	Teilweise (Hit Validation)	Nein	Nein
Experte 3	Ja (Vektoren, Quaternionen, Physics)	Implizit	Ja (State Machine, Decision Trees)	Nein	Nein
Experte 4	Nein (nicht explizit)	Ja (A*)	Ja (Finite State Machine, Behaviour Tree)	Nein	Nein
Experte 5	Ja(Hinderniserkennung)	Ja (Navmesh)	Ja (Behavior Trees)	Nein	Nein
Experte 6	Ja (Bewegung, Combat States)	Ja (Navmesh)	Ja (Behavior Trees)	Nein	Nein
Experte 7	Ja (Movement+Physics)	Ja (Physics+Movement)	Ja (State Machines, Dialoge, Animation)	Nein	Teilweise (LLM-Integration, API)
Experte 8	Indirekt (über Pathfinding)	Ja	Ja (Bots, Decision Making)	Nein	Nein
Experte 9	Ja (viel gemacht)	Ja	Ja (Decision Making, Tactical/Strategic)	Ja	Ja (früher, lernende KI, aber selten in Games)
Experte 10	Ja (Tilemap Movement)	Ja (A*)	Ja (State Machines, Behavior Trees)	Ja Teilweise (Strategische Heuristiken)	Ja (Uni-Projekt Racing AI, selten)

Experte	Procedural Content Generation	Board Games	Sorting	Physics
Experte 1	Nein	Ja (Min-Max, Decision Making)	Nein	Teilweise (Movement Prediction)
Experte 2	Ja (Heightmap, Terrain)	Nein	Ja	Ja (Server/Client Validation)
Experte 3	Nein	Nein	Nein	Ja (Quaternionen, Vektoren)
Experte 4	Ja (World/Landscape)	Nein	Nein	Ja Eher KI, aber auch genutzt
Experte 5	Nein	Nein	Nein	Ja (Rigid Body, Hinderniserkennung)
Experte 6	Nein	Nein	Nein	Ja (Rigid Body, Combat Status)
Experte 7	Nein	Nein	Nein	Ja (Physics, Gravitation)
Experte 8	Ja (Wave Function Collapse)	Nein	Ja	Nein
Experte 9	ja Wenig (selten gebraucht)	Ja (MinMax, eigene AIs für Brettspiele)	Ja (Bubble Sort im MMO)	Ja (eigene Physikmodelle, Movement)
Experte 10	Ja (Genre-abhängig)	Ja (Minimax privat/Uni, nicht kommerziell)	Nein	Ja (Tilemap, Geometry)