Maximilian Eibl, Martin Gaedke (Hrsg.): Informatik 2017: CAAI4Games,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1

Ausprägung von Verhaltensprofilen zur Navigation für KI-Agenten in Unity®

Christopher Käding, Holger Langner, Manuel Heinzig, Sebastian Schleier,
Christian Roschke, Rico Thomanek und Marc Ritter [1]

Abstract: Dieser Beitrag untersucht das Verhalten eines klassischen Wegfindungsalgorithmus in einigen künstlich erzeugten Szenarien. Die Umsetzung geschieht mittels der Game-Engine Unity® und deren grundlegendem Navigationsobjekt NavMesh, auf welchem sich ein Agent unter Verwendung einer modifizierten Version des A*–Algorithmus bewegen und selbstständig seinen Weg durch ein aufgebautes Labyrinth finden kann. Für die Evaluation erfolgt eine Parametrisierung verschiedener Agenten, sodass diese in einem für den Kontext passenden Videospiel als an verschiedenen, menschenähnlichen Persönlichkeitseigenschaften orientierte KI-Gegner agieren können. Getestet werden sie auf einer Anzahl konzipierter Strecken mit bewusst herausfordernden Charakteristiken, um die Fähigkeiten der Agenten in Extremsituationen zu erproben. Es zeigt sich, dass der beim subjektiven Betrachten erweckte Eindruck über die Qualität der Durchläufe mit den in den angewendeten Metriken abgebildeten Besonderheiten korreliert. Außerdem offenbart der Wegfindungsalgorithmus in Grenzsituationen Auffälligkeiten in den Bewegungspfaden. Auf Grundlage der gewonnenen Erkenntnisse über Konzeption, Umsetzung und Bewertung der Experimente können zukünftig ähnliche Szenarien mit überschaubarem Aufwand untersucht und evaluiert werden.

Keywords: Künstliche Intelligenz, A*, Wegfindung, Agentenprofile, Unity®

1 Motivation

In den vergangenen Jahren hat sich bei der Kreation von Videospielen die Verwendung so genannter Game-Engines [Gr09, S. 11 f.] zum Standard herausgebildet. Diese geben dem Entwickler einen Grundstock von Werkzeugen an die Hand, mit deren Hilfe sie auf relativ einfachem Weg komplexe Welten erschaffen und die benötigten Interaktionen und Logiken einarbeiten können. Teil der Basisfunktionalitäten sind zumeist auch einfache Versionen von Algorithmen aus dem Bereich der Künstlichen Intelligenz (KI), welche computergesteuerten Gegnern ein Mindestmaß an selbstständiger Handlungsweise ermöglichen. Weiterführende Methoden zeichnen sich im wesentlichen durch den Einbezug eines größeren Parameterbereiches aus, was in einer fortschrittlichen Entwicklungsstufe das adaptive Lernen von Spielerverhalten und entsprechende eigenständige Anpassungen des KI-Gegners in Abhängigkeit von den Leistungen des Spielers erlauben soll. Ein solch individuell angepasstes Verhalten wird aufgrund der Komplexität der unterliegenden Algorithmik in der Praxis nur selten umgesetzt.

[1] Hochschule Mittweida, Professur Medieninformatik, Technikumplatz 17, D-09648 Mittweida, marc.ritter@hs-mittweida.de

An dieser Stelle setzt der vorliegende Beitrag an. Im Folgenden wird ein Vorgehensmodell [Sc82, Kap. 2] für die Untersuchung verschiedener Verhaltensausprägungen von variabel parametrierten KI-Agenten am Beispiel des in der Game-Engine Unity® [Unity17] bereitgestellten implementierten Wegfindungsalgorithmus abgeleitet. In Anlehnung an reale Spielszenarien, exemplarisch eine Verfolgungsjagd durch ein Labyrinth oder ein klassisches Arcade-Rennspiel, wird eine Reihe semantischer Verhaltensprofile erstellt und anhand thematisch passender Metriken eruiert. Die so gewonnenen Erkenntnisse und die entstandene Verfahrenskette sind ein erster, grundlegender Schritt zu einer automatisierten Optimierung der in den KI-Algorithmen verwendeten Parametrierungen und das daraus resultierende Verhalten.

2 Grundlagen

Die Implementierung der parametrisierten KI-Agenten sowie der passenden Strecken finden direkt in der Unity®-Software statt, da sie die benötigten Funktionalitäten bereits ab Werk bereitstellt. Verwendete Komponenten mit zentraler Bedeutung für die Navigation durch eine unbekannte digitale Strecke sind das *NavMesh* [NM17] und der *NavMesh-Agent* [NMA17].

Unity® ist eine Laufzeit- und Entwicklungsumgebung für Spiele des Unternehmens *Unity Technologies* mit Hauptsitz in San Francisco. Zur Erstellung der Umgebung und den Agenten werden in diesem Beitrag ausschließlich 3D-Komponenten von Unity® verwendet. Nachfolgende Experimente werden mit einer Bildrate von ca. 50 fps abgetastet. Dadurch ergibt sich ein Abstand von etwa 0,02 Sekunden zwischen zwei aufeinanderfolgenden Messzeitpunkten.

Das Suchverfahren A* [HNR68] bietet eine optimale Möglichkeit der Wegfindung, ist in seiner ursprünglichen Form jedoch aus Performanzgründen für eine 3D-Umgebung und dem dafür notwendigen extrem feingranularen Graph nicht sonderlich geeignet. Um dieses Problem zu lösen, werden die Knoten durch sinnvoll platzierte geometrische Figuren (Polygone) ersetzt [CS11].

Die Entwicklungsumgebung bietet dem Nutzer weiterhin die Möglichkeit, das *NavMesh* an einen Agenten anzupassen. Das erstellte Mesh kann so bei der Navigation die Außenmaße des Agenten ebenso berücksichtigen, wie die maximal erklimmbare Steigung und die Höhe begehbarer Stufen. Der elementare Parameter *Radius* beschreibt dabei den für den Agenten reservierten Raum, in welchen kein anderes Objekt eindringen kann. Ein Agent mit großem Radius kann sich folglich nicht durch einen entsprechend engeren Bereich bewegen. Analog dazu existiert der Parameter *Height*, welcher die minimal notwendige Durchgangshöhe unter einem Hindernis festlegt.

Der Agent ist eine Komponente, welche das NavMesh abtastet und sich auf ihm fortbewegt. Er kann das Mesh nicht verlassen und ist gezwungen sich bei der Wegfindung an seine algorithmische Logik zu halten. Darüber hinaus ist er an keine physikalischen Regeln

gebunden, sondern wird im Raum frei durch Manipulation der Ortskoordinaten verschoben. Die folgenden Parameter sind für die Bewegung relevant.

Acceleration: Die Acceleration gibt sowohl die Befähigung des Agenten zu beschleunigen, als auch zu bremsen an. Sie ist in Unity-Maßeinheiten pro Sekunde im Quadrat angegeben. Unmittelbar daraus ergibt sich der Radius einer zu fahrenden Kurve bei gegebener Geschwindigkeit.

Speed: Dieser Wert beschreibt die maximale Geschwindigkeit des Agenten in Unity-Maßeinheiten/Sekunde. In Abhängigkeit von der Acceleration kann das Erreichen der Höchstgeschwindigkeit eine lange und gerade Strecke bedingen.

Quality: Der Agent versucht je nach Einstellung (none, low, medium, good & high quality) die Kollision mit Hindernissen und anderen Agenten vorsorglich zu vermeiden. Die Qualität gibt hierbei an, wie viel CPU-Zeit in die Berechnung einer möglichst kollisionsfreien Route fließt. Je höher die Qualität, desto näher bewegt sich der Agent an einem Hindernis vorbei. Bei der Einstellung von *none* ignoriert der Agent andere Agenten und Hindernisse und richtet die Route ausschließlich nach den Vorgaben des *NavMeshs*.

3 Vorgehensweise

Das geplante Vorgehen basiert auf zwei Hauptpfeilern. Zum einen muss ein Szenario entstehen, in welchem sich die Agenten bewegen können. Zum anderen sind die Agenten selbst in einer ökonomischen Art und Weise in der Software umzusetzen und einzubinden. Diese beiden groben Oberpunkte werden durch Anforderungen und perspektivische Konzepte weiter spezifiziert, sodass die endgültige Vorgangsbeschreibung aus vier Hauptteilen besteht.

Modellierung der Szene: Aus vorher klar definierten Grundbausteinen entstehen einige Strecken mit verschiedenen Charakteristiken, auf denen sich der Agent eigenständig vom Start zu einem Zielpunkt bewegen soll. Für zufriedenstellende Testergebnisse müssen die Strecken gewissen Basisansprüchen genügen: (i) Sie soll eine plausible Ideallinie beinhalten, die von jedem Agent gleichermaßen bewältigt werden kann. (ii) Sie soll Unterschiede und Besonderheiten zwischen den Verhaltensprofilen der Agenten zur Geltung kommen lassen. (iii) Sie soll klar definierte Anhaltspunkte zum Auslesen numerischer Werte für eine spätere statistische Auswertung im Bezug auf die von den Agenten zu lösende Aufgabe bereitstellen. Grundsätzlich soll eine Strecke also schematisch klar strukturiert und für jeden Agent ausführbar sein und mindestens einen Start- und Endpunkt enthalten, um an diesen fest definierte Informationen abrufen zu können.

Entwicklung eines prototypischen Agenten: In einer ersten Iteration ist ein Agent auf die einfachst mögliche Art und Weise zu implementieren. Auf einer sorgfältig konzeptionierten Strecke kann sich dieser nach grundlegenden Regeln vom Start zum Ziel bewegen. Seine Umsetzung unterliegt den folgenden Anforderungen: (i) Die Entwicklung muss unmittelbar mit den im jeweiligen Tool vorhandenen Werkzeugen möglich sein. (ii) Die notwendige

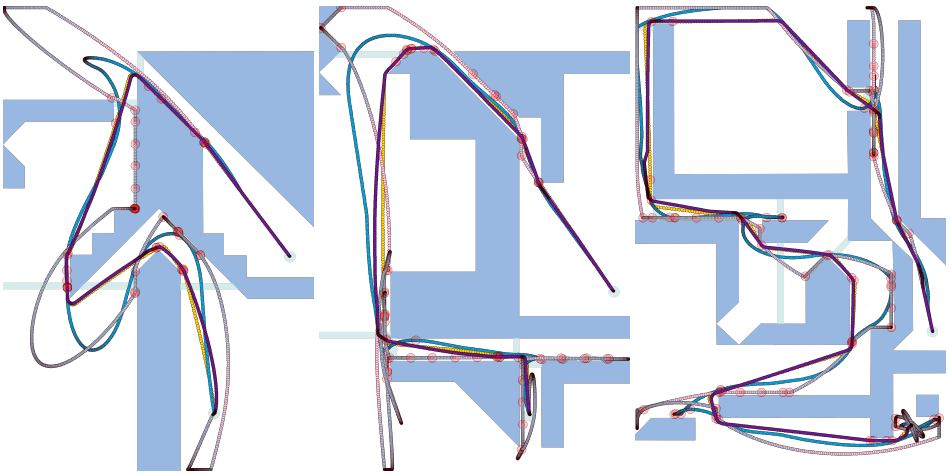

Abb. 1: Typische Wegeverläufe der vier definierten Agententypen für die drei Strecken: Track 1 (links), Track 2 (Mitte) und Track 3 (rechts).

intellektuelle Anpassung der Verhaltensparameter soll einfach realisierbar sein. Die verwendete Game-Engine stellt für diese Zwecke ein einfaches Navigationskonzept bereit, dessen Anwendung sich in den generellen Arbeitsfluss von Unity® einbettet.

Experimente mit prototypischer Parametrisierung: Nach Erstellung aller notwendigen Softwarebestandteile können die ersten, grundlegenden Experimente durchgeführt werden. Die genutzten Parameter sind dabei entweder auf vorgegebene Standardwerte oder manuell ermittelte Größen gesetzt. Der vorliegende Beitrag skizziert dieses Verfahren und dokumentiert die damit festgestellten praktischen Ergebnisse.

Experimente mit automatischer Parameteranpassung: Als endgültige Iteration soll der Agent in der Lage sein, sein Verhalten selbständig anzupassen. Diese Variante erfordert tiefgreifendes mathematisches Fachwissen und umfassende Programmierfertigkeiten und kann daher im Rahmen des vorliegenden Beitrags nur ausblicksartig inspiriert werden.

Die aufgeführten Punkte skizzieren das Vorgehen für einen iterativen Entwicklungsprozess, der im Detail an den jeweiligen konkreten Projektkontext angepasst werden kann. Im Fokus dieser Publikation steht als Machbarkeitsstudie die Umsetzung der ersten drei Vorgehensschritte. Es wird dazu eine funktionsfähige Softwaregrundstruktur vorgestellt, die die Durchführung einfacher Experimente mit mehreren individuell parametrisierten KI-Agenten ermöglicht und erste Ergebnisse betrachtet.

4 Experimentelles Setup

Für die Durchführung der Experimente sind drei Strecken vorgesehen. Diese sind jeweils aus einem Raster von 14 x 21 Einheiten aufgebaut, deren Kantenlänge einer Unity-Maßeinheit entspricht. Als Hindernisse dienen Formen unterschiedlich aufgeteilter Einheitsquadrate: Ein vollständig solider Block mit einer Kantenlänge von einer Unity-Maßeinheit und ein halber Block, wobei dessen Trennlinie entlang der Diagonale stattfindet, sodass Einteilungen im 45-Grad-Winkel möglich sind. Mittels dieser Formvariationen lassen sich die im nachfolgenden Abschnitt beschriebenen Strecken im genormten Raster erzeugen.

4.1 Strecken

Die generierten Strecken sind in drei unterschiedliche Schwierigkeitsgrade unterteilt. Hierfür wurden folgende Bedingungen festgelegt:

Track 1 (einfach): Die erste Strecke soll keine große Herausforderung für die einzelnen Agenten darstellen. Zum Manövrieren in den Kurven sind große Auslaufzonen vorhanden. Weiterhin gibt es keine alternativen Wege und Abzweigungen, wobei selbst eine Kollision mit der Wand weniger schwerwiegend erscheint.

Track 2 (mittel): Die zweite Strecke soll für bestimmte Agententypen erste Herausforderungen bieten. Hier existiert ein alternativer Zielweg und Fehler beim Verpassen von Abzweigen werden durch zeitaufwändige Wendemanöver bestraft.

Track 3 (schwer): Die letzte Strecke stellt eine große Herausforderung für die Agenten dar. Die Maximalgeschwindigkeit ist seltener erreichbar, da häufige und enge Kurven die Beschleunigung erschweren. Ferner stellen Abzweigungen die Agenten vor Entscheidungen.

Bei allen Strecken startet das Tracking der Agenten erst nach einer vordefinierten Initialisierungszeit von einer Sekunde, in der jegliche Bewegungen ausgeschlossen sind. Das Tracking ist zu Ende, sobald der Agent das Ziel erreicht und am Zielpunkt eine Geschwindigkeit von Null aufweist.

4.2 Profile und Parametrierungen

In Anlehnung an menschliche Charakterzüge [Sc05] erhalten die virtuellen Agenten anhand weniger Einstellungen einen jeweils eigenen Fahrstil. Die dafür notwendigen Parametervariationen werden im Folgenden erläutert.

Gemeinsame Parameter

Die Beschreibung der Standardparameter findet im Abschnitt 2 statt. Dabei fällt auf, dass der den Agenten zugewiesene Radius kleiner ist als der verwendete *Box Collider*. In Zusammenspiel mit den unterschiedlichen geometrischen Formen (Zylinder und Würfel) ist es somit möglich, dass der Agent selbst dann mit einem Hindernis kollidiert, wenn er eine hohe Kollisionsvermeidung verwendet. Außerdem nutzen alle Agenten die Komponente *Rigidbody* um Kollisionen zu erkennen. Damit die weiteren Anwendungsfelder von *Rigidbody* nicht die planmäßige Durchführung der Streckendurchläufe verhindern, erhalten sie jedoch eine Masse von annähernd Null zugewiesen.

Individuelle Parametrisierung

Einige Parameter sind von Agent zu Agent unterschiedlich und verleihen ihnen ihre einzigartige Persönlichkeit. Die Tabelle 1 gibt eine Übersicht über alle variierten Einstellungen.

Agententyp	Höchstgeschwindigkeit	Beschleunigung	Kollisionsvermeidung
Safe	15	50	high
Neutral	30	20	medium
Risky	60	20	none
Cheater	60	1000	none

Tab. 1: Parameter der einzelnen Agenten

Die Parametrisierungen wurden so gewählt, dass sich vier verschiedene Agententypen ergeben. Diese weisen jeweils ein charakteristisches Bewegungsverhalten auf, um idealtypische, gut unterscheidbare Skill-Levels oder Spielcharaktere aus Sicht eines menschlichen Spielers zu repräsentieren.

- **Agent *Safe* (lila):** Dieser Agent bewegt sich sehr langsam durch die Strecke. Er versucht, Kollisionen und überflüssige Wege zu vermeiden, sollte also die Ideallinie finden.
- **Agent *Neutral* (türkis):** Der ausgeglichene Agent bildet einen Kompromiss zwischen dem Finden einer hohen Geschwindigkeit und dem Vermeiden überflüssiger Wege. Dadurch kann er gelegentlich aus der Kurve getragen werden.
- **Agent *Risky* (grau):** Bei dem riskanten Profil fährt der Agent sehr schnell, wodurch er allerdings häufig von der optimalen Kurve abweichen und Kollisionen verursachen wird.
- **Agent *Cheater* (gelb):** Dieser Agent ist eigentlich nicht schlagbar. Er fährt nahe der Idealline mit einer sehr hohen Geschwindigkeit und gleichzeitig großer Beschleunigung, wodurch sein Wendekreis sehr klein bleibt.

Implementierung

Eine Beschreibung des in Unity® verwendeten Navigationssystems findet in Abschnitt 2 statt. Zusätzlich zu den bereits vorhandenen Funktionen werden verschiedene, problemspezifische Skripte implementiert. Dabei modelliert ein Skript das Agentenverhalten; ein weiteres zeichnet folgende relevante Bewegungsdaten des Agenten in einer Log-Datei auf:

- **Position:** Die Fläche, auf der sich der Agent fortbewegt, wird mit einem definierten Raster versehen. In unserem Fall entspricht dies 1400 x 2100 Einheiten.
- **Kollisionen:** Bei der Kollision des Agenten mit einem anderen Objekt schreibt das Skript den Position des Hindernisses heraus. Da keine physikalischen Außenbegrenzungen der Strecken mit Hindernisobjekten modelliert wurden, findet an Rand des *NavMeshs* zwar eine Begrenzung der Bewegung, aber keine Kollisionsdetektion statt.
- **Geschwindigkeit:** Die Geschwindigkeit, ermittelt durch die Differenz zwischen der aktuellen Position und der Position zum Zeitpunkt des letzten Frames.Sie verwendet als Einheit die *Unity-Maßeinheit pro Frame*; die näherungsweise Umrechnung in *Unity-Maßeinheit pro Sekunde* erfordert die Multiplikation mit der Abtastrate von 50 Hz.

5 Evaluation

Im Zuge der Evaluation sollen die aus den gewählten Parametrierungen resultierenden Bewegungsprofile näher untersucht werden. Von Interesse ist dabei zum einen, welche qualitativen Varianten von Bewegungsmustern mit den sehr einfachen Beschreibungsmitteln zur Agenten- und Streckenmodellierung überhaupt erzielt werden können. Zum zweiten ist zu bewerten, ob sich charakteristische Unterschiede zwischen den gewählten Agententypen ausprägen und in welchem Maße diese mit quantitativen Metriken statistisch zuverlässig erfassbar sind, um die Grundlage für eine spätere automatische Parameteranpassung zu legen.

5.1 Vorgehen zur Evaluation

Im Rahmen der Evaluation werden die Auswirkungen der individuellen Parametrierungen auf das Agentenverhalten untersucht. Dazu wurden für jeden Agententyp und für jede Strecke mehrere Durchläufe realisiert und dabei mit Hilfe des Loggingskripts der jeweils zurückgelegte Bewegungspfad vollständig aufgezeichnet.

Um Gemeinsamkeiten und Unterschiede zwischen allen Durchläufen in quantitativen Statistiken zu erfassen, wurden anschließend pro Durchlauf folgende Metriken erhoben:

Rundenzeit, zurückgelegte Weglänge, Anzahl Kollisionen in einem Durchlauf sowie das Histogramm des Geschwindigkeitsprofils.

Zum Zweck der Auswertung des Agentenverhaltens nach qualitativen Merkmalen wurden einzelne Datenattribute in eine zweidimensionale Heatmap-Darstellung integriert, die eine Draufsicht auf die Strecke repräsentiert. Für die qualitative Untersuchung sind vor allem folgende Fragestellungen von Interesse:

- Führen die verschiedenen Parametrierungen zu qualitativ unterschiedlichen Varianten von Bewegungsmustern und werden diese aus Sicht eines menschlichen Spielers auch als solche wahrgenommen?
- Resultieren aus der Umsetzung des Simulationsmodells in Unity® Besonderheiten im Agentenverhalten oder gar unplausible Bewegungsmuster?

Die Ergebnisse der Evaluation sollen dann eine Abschätzung ermöglichen, ob die Modellierungsmöglichkeiten ausreichend sind, um KI-Agenten mit plausiblen und als unterschiedlich wahrnehmbaren Fähigkeitsstufen zu entwickeln.

5.2 Resultierendes Agentenverhalten

Dieser Abschnitt fasst die Ergebnisse der quantitativen und qualitativen Untersuchungen zusammen, und stellt zu diesem Zweck die eingangs gewählten vier Agententypen einander gegenüber.

Quantitative Auswertung des Agentenverhaltens

Im Ergebnis mehrerer Durchläufe ergibt sich die statistische Gegenüberstellung in Tabelle 2.

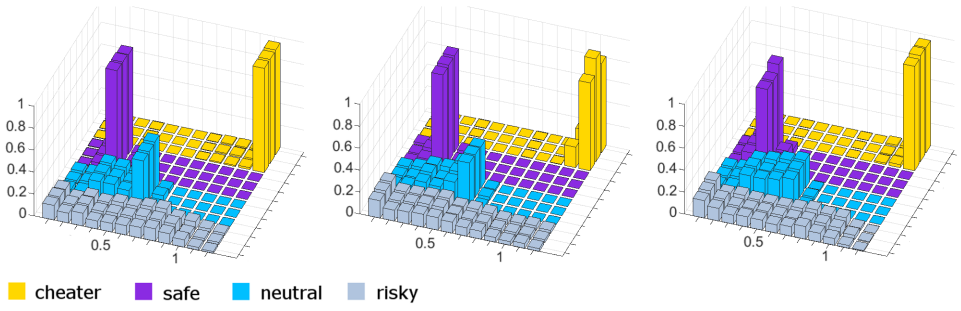

Abb. 2: Histogrammdarstellung der Geschwindigkeitsprofile aller Agententypen und Strecken

Messgröße	Strecke	Cheater	Neutral	Risky	Safe
Streckenzeit (μ, σ)	1	5.98 (0.01)	19.54 (0.10)	25.19 (2.26)	25.32 (0.18)
Kollisionen (μ, σ)	1	10.33 (0.58)	3.00 (1.73)	24.67 (4.04)	13.00 (1.73)
Wegstrecke (μ, σ)	1	3479 (3.16)	4349 (4.36)	6396 (386.92)	3482 (1.20)
Streckenzeit (μ, σ)	2	6.57 (0.03)	21.47 (0.06)	46.12 (1.80)	28.12 (0.11)
Kollisionen (μ, σ)	2	10.67 (0.58)	8.67 (1.53)	54.00 (10.39)	10.67 (0.58)
Wegstrecke (μ, σ)	2	3737 (2.55)	4494 (39.02)	10316 (303.44)	3774 (6.81)
Streckenzeit (μ, σ)	3	10.5 (0.02)	40.69 (0.07)	39.98 (0.22)	45.95 (0.15)
Kollisionen (μ, σ)	3	14.67 (1.15)	24.67 (1.53)	57.33 (18.50)	10.33 (0.58)
Wegstrecke (μ, σ)	3	6203 (10.76)	7821 (11.99)	9514 (52.46)	6212 (2.94)

Tab. 2: Quantitative Auswertung des Agentenverhaltens für drei Strecken

Die Agenten *Cheater* und *Safe* markieren augenscheinlich die gegenüberliegenden Skalenendpunkte an potentiellen Ausprägungen von Bewegungsmustern, die gemeinsam weitgehend einer Ideallinie folgen. Dabei sind die Fähigkeiten eines menschlichen Spielers im Laufe seiner Fähigkeitsausprägung am ehesten knapp oberhalb des Agenten *Neutral* oder zwischen diesem und Agent *Safe* einzuordnen, während der Agent *Cheater* einen nicht erreichbaren Extremwert bezüglich der schnellsten Rundenzeiten repräsentiert. Agent *Risky* fügt dieser Bandbreite an Fähigkeitsstufen eine weitere Variante hinzu: Er ist über weite Strecken ein sehr herausfordernder Gegner, der auch den Agenten *Neutral* vereinzelt schlagen kann. Jedoch ist er stark fehleranfällig, so dass sich für einen menschlichen Spieler im Streckenverlauf Überholmöglichkeiten ergeben.

Eine eindeutige quantitative Abgrenzung aller vier Agententypen ist auf der Grundlage der in Tabelle 2 erfassten Werte allein jedoch nicht möglich. Daher wurden die Geschwindigkeitsprofile aller Agenten und Durchläufe in normierte Histogramme übertragen. Abbildung 2 zeigt für jedes der drei Streckenprofile einige ausgewählte Geschwindigkeitshistogramme. Die waagerechte Achse gibt die Geschwindigkeit, und die senkrechte Achse den zugehörigen Häufigkeitsanteil dieser Geschwindigkeit im jeweiligen Durchlauf an. Alle vier Agententypen wurden (markiert mit der zugehörigen Farbe) hintereinander aufgestaffelt, und pro Agent drei exemplarische Durchläufe für die Abbildung ausgewählt.

Daraus lässt sich abschätzen, dass bereits die individuelle Parametrisierung mit einer Out-of-the-Box-Methode zu Agententypen geführt hat, die anhand der Charakteristika ihrer Geschwindigkeitsverteilung zuverlässig statistisch identifizierbar sind. Die einzelnen Parametrierungen unterscheiden sich sowohl in der Ausprägung des globalen Maximums der Geschwindigkeitsverteilung, als auch in den Varianzen sehr eindeutig voneinander. Der nachfolgende Abschnitt führt dies im Zusammenhang mit der qualitativen Untersuchung der Bewegungsmuster näher aus.

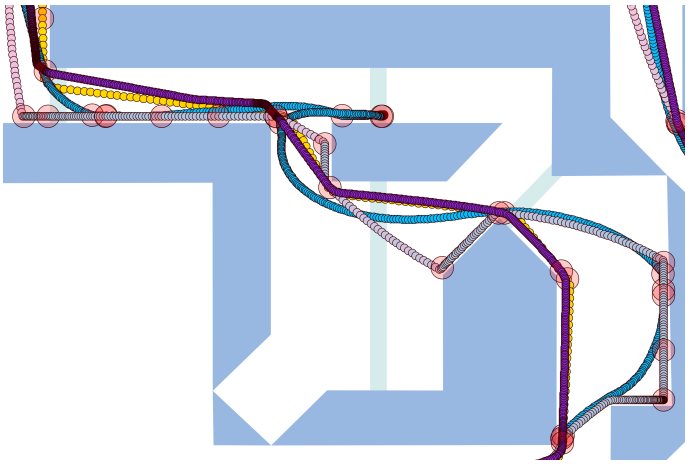

Abb. 3: Charakteristische Bewegungspfade aller vier Agenten.

Beobachtbare Varianten im Agentenverhalten

Abbildung 3 überblendet charakteristische Bewegungspfade der vier Agententypen innerhalb desselben Streckenausschnitts aus Strecke 3, um die Bewegungscharakteristiken zu veranschaulichen.

Agent *Safe* (im Bild violett) folgt immer der kürzesten Wegstrecke bis zum Ziel und fährt dabei auf Grund seiner geringen Endgeschwindigkeit alle Ecken von Hindernissen direkt an. Auch kann er von seiner Endgeschwindigkeit auf kurzer Strecke so stark verzögern, dass er ohne signifikanten Seitenabstand direkt um eine Hinderniskante herum navigieren kann. Der Agent bewegt sich in seinem Geschwindigkeitsprofil fast immer konstant mit der gewählten Endgeschwindigkeit, die wesentlich unter der aller anderen Agententypen liegt. Der Einfluss des gewählten Setups für die Hindernisvermeidung ist an diesem Agententyp ebenfalls gut nachvollziehbar. Es finden von Zeit zu Zeit Kollisionen mit Hindernisecken, aber vereinzelt auch mit geraden Hinderniskanten statt, da die Empfindlichkeit des vorsorglichen Kollisionsvermeidungsverhaltens gegenüber der Kollisionsdetektion in allen vier individuellen Parametrierungen sehr weit herabgesetzt wurde.

Agent *Neutral* (im Bild blau) fährt auf Grund seiner höheren Endgeschwindigkeit größere Kurvenradien und kollidiert dadurch von Zeit zu Zeit mit Hinderniskanten, die dem Bewegungskorridor gegenüberliegen, so dass eine aufwendige Umkehrbewegung auf den Zielkurs notwendig wird. Solange keine derartigen Kollisionen auftreten, kann der Agent innerhalb eines hohen Anteils im Geschwindigkeitsprofil eine hohe Endgeschwindigkeit beibehalten.

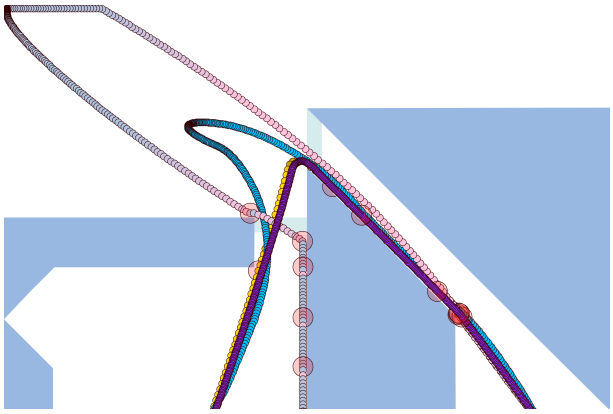

Abb. 4: Charakteristisches Kurvenverhalten der vier Agenten.

Agent *Risky* (im Bild grau-magenta) fährt durch seine zu hohe Endgeschwindigkeit Kurven so weit aus, dass er häufig sowohl mit gegenüberliegenden Hinderniskanten als auch mit den äußeren Streckenbegrenzungen kollidiert, wie der vergrößerte Ausschnitt aus Strecke 1 in Abbildung 4 zeigt.

Die Gesamtzahl an Kollisionen über alle Streckenverläufe hinweg liegt bei Agent *Risky* deutlich über der aller anderen Agententypen. Auch verlässt er die optimale Wegstrecke dabei soweit, dass er über mehrere Elemente eines *NavMesh* hinweg umkehren muss, um diese wieder zu finden. Beim Umkehren kann er dabei in eine Pendelbewegung geraten, in deren Verlauf er die richtige Abzweigung zum Ziel mehrmals verfehlt. Dieses Pendeln tritt häufig auch um den Zielpunkt herum auf, was sich für die Spielszene als unplausibel erweist und durch eine alternative Modellierung des Ziels im Streckendesign (z.B. Zielstrich anstelle eines Kollisionspunktes) behoben werden sollte. Diese Pendelbewegungen lassen sich anhand von Abbildung 5 nachvollziehen (der Zielpunkt ist hier jeweils als türkisfarbener Kreis dargestellt).

Agent *Risky* übertrifft in einzelnen Streckenabschnitten die Endgeschwindigkeit des Agenten *Neutral* signifikant. Die häufigen Kollisionen beeinträchtigen seine Navigation allerdings so stark, dass diese hohe Endgeschwindigkeit nur über geringe Anteile im Geschwindigkeitsprofil hinweg überhaupt erreicht werden kann. Stattdessen treten im Geschwindigkeitsprofil nun Bereiche auf, in denen die Momentangeschwindigkeit sogar noch unter der des Agenten *Safe* liegt.

Agent *Cheater* (in den Abbildungen gelb) generiert einen beinahe identischen Pfad wie Agent *Safe* d.h. ist in der Lage, den kürzestmöglichen Pfad mit einer konstanten Geschwindigkeit zu verfolgen, die diejenige aller anderen Agententypen weit übertrifft. Etwaige Kollisionen haben dabei keinen Einfluss, da der Agent auf Grund seiner hohen Beschleunigung seine Höchstgeschwindigkeit innerhalb eines Simulationszeitschrittes wieder erreichen kann. Das

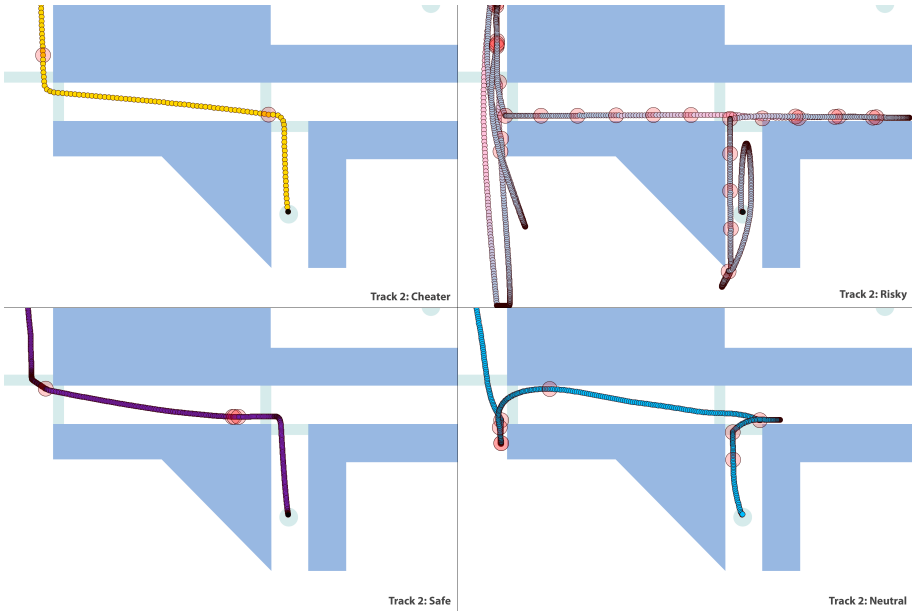

Abb. 5: Detailausschnitt aus charakteristischen Bewegungspfaden.

Verhalten zur Kollisionsvermeidung ist in der Parametrierung abgeschaltet worden. Dadurch kommen auch lokale Unterschiede zum Navigationspfad des Agenten *Safe* zustande.

Auffällige Bewegungsmuster

Anhand der grafischen Darstellung einzelner Bewegungspfade werden Besonderheiten im Agentenverhalten augenfällig, die sich aus der Referenzdokumentation für Unity®-Entwickler nicht direkt ableiten lassen, sondern einer weitergehenden Eruierung des zugrundeliegenden dynamischen Simulationsmodells bedürfen.

So ist etwa Agent *Safe* in der Lage, einer geraden Hinderniskante exakt parallel zu folgen, während die Trajektorie von Agent *Neutral* minimal hin und her pendelt. Auch lässt sich häufig beobachten, dass das Hindernisvermeidungsverhalten in der Regel nicht sofort einen optimalen Ausweichkurs generiert (wie z.B. den direkten Kurs zum Zielpunkt, falls eine Sichtlinie existiert). Stattdessen resultiert meist eine davon abweichende Ausweichrichtung, augenscheinlich weil eine weitere nicht näher spezifizierte Vektorkomponente in die Berechnung einfließt.

6 Zusammenfassung und Ausblick

Es stellt sich heraus, dass bereits mit den in Unity® verfügbaren Möglichkeiten signifikant divergente Profile für einzelne Agenten modelliert werden können. Deren Bewegungsmuster bleiben in den Versuchen über mehrere verschiedene Strecken weitgehend ähnlich. Dieser Zusammenhang lässt sich durch qualitative und quantitative Metriken konkret beschreiben. Dieser Punkt bildet eine wichtige Grundlage für zukünftige Forschungsarbeiten zur Automatisierung des Vorgangs zur Parameteranpassung an gewünschte Fähigkeitsstufen.Ferner wäre als ein künftiger Analyseschritt zu klären, wie die beobachteten Auffälligkeiten konkret von der Implementierung des Hindernisvermeidungsverhaltens in Unity® bedingt werden.

Literatur

[CS11] Cui, X.; Shi, H.: A*-based Pathfinding in Modern Computer Games. International Journal of Computer Science and Network Security 11/1, S. 125–130, 2011.

[Gr09] Gregory, J.: Game Engine Architecture. CRC Press, 2009.

[HNR68] Hart, P. E.; Nilsson, N. J.; Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics 4/2, S. 100–107, Juli 1968.

[NM17] Unity Technologies, 2017, URL: https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html, Stand: 07. 06. 2017.

[NMA17] Unity Technologies, 2017, URL: https://docs.unity3d.com/Manual/class-NavMeshAgent.html, Stand: 07. 06. 2017.

[Sc05] Schmidt, H.: Temperamentenlehre (Neuzeit). In: Enzyklopädie Medizingeschichte. De Gruyter, Berlin/ New York, 1382 f. 2005.

[Sc82] Schmidt, G.: Grundlagen der organisatorischen Gestaltung. C.E. Poeschel Verlag, Stuttgart, 1982.

[Unity17] Unity Technologies, 2017, URL: https://docs.unity3d.com/Manual/, Stand: 07. 06. 2017.