Bordeaux 2023

Der für Bordeaux entwickelte Taskmanager stellt die zweite Iteration dar. Anstatt eines einzigen Rule-Systems werden hier mehrere unterschiedliche Module verwendet. Zunächst wird vom Rolemanager jedem Roboter eine Rolle zugewiesen. Für jede Rolle existieren einzelne Verhaltensweisen, wie beispielsweise die Flexwall. Außerdem wurden zur Gestaltung der Offensive auch erstmals MinMax-Bäume eingesetzt, um Pässe zu planen.

Übersicht

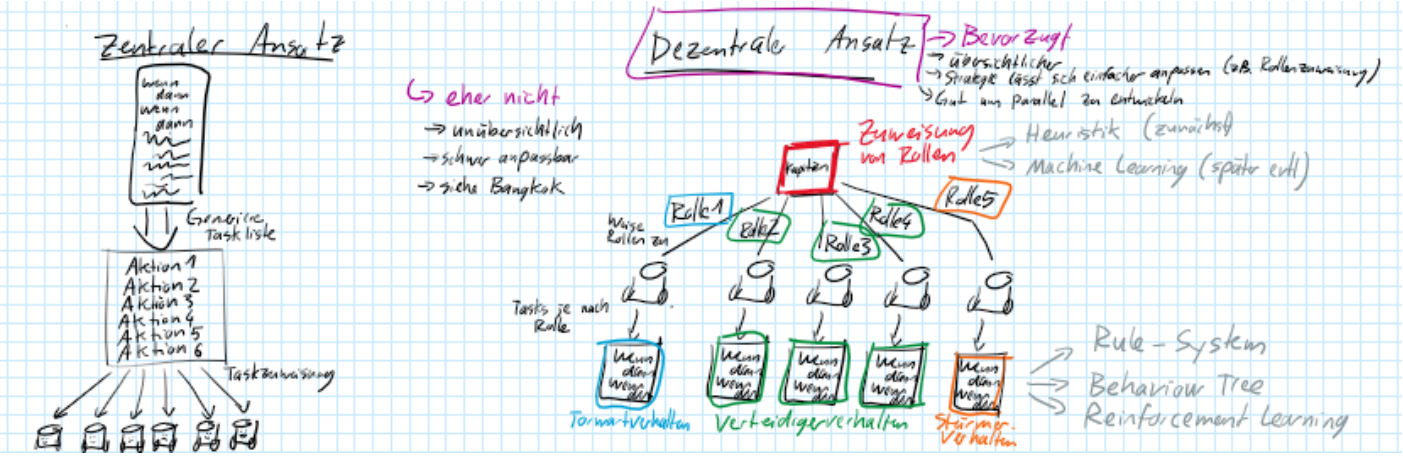
Der Bordeaux-Taskmanagers stellt die zweite Iteration der Strategieentwicklung dar. Der Taskmanager Bangkok wurde aufgrund seiner Probleme hinsichtlich Komplexität und schlechter Erweiterbarkeit nach dem Wettbewerb komplett verworfen. Stattdessen wurden aus den Erfahrungen die Anforderungen für die Entwicklung eines neuen Systems abgeleitet, welches sich vor allem auf Erweiterbarkeit sowie Einfachheit in der Entwicklung fokussiert.

Grundprinzip und Architektur

Unter Taskmanager Bangkok wurde der ursprüngliche Ansatz beschrieben, die Strategie über ein einziges *Rule-System* zu realisieren. Aus den dort beschriebenen Problemen wurden Anforderungen für eine neue Architektur abgeleitet, die die Entwicklung, Anpassbarkeit und Erweiterung des Systems vereinfachen sollen. Die Ziele lassen sich folgendermaßen klassifizieren:

- *Mehrschrittiger Entscheidungsprozess*; Hiermit soll eine bessere Kapselung der Module erreicht werden, wodurch diese unabhängig voneinander entwickelt werden können. Außerdem soll nach dem "Teile und Herrsche"-Prinzip das große Problem der Strategie in mehrere Teilprobleme aufgeteilt werden, um die Lösbarkeit des Problems zu vereinfachen. Der wesentliche Ablauf der Task-Zuweisung lässt sich folgendermaßen beschreiben:
  1. Analyse des Spielfeldes
  2. Festlegung der Rollenverteilung (Anzahl Angreifer/Verteidiger...)
  3. Zuweisung der Rollen je nach gewünschter Rollenverteilung und Roboterposition
  4. Festlegung der auszuführenden Tasks in Abhängigkeit der Rollenverteilung
  5. Zuweisung der Tasks zu den verfügbaren Robotern in Abhängigkeit der Rollenverteilung
- *Kapselung der Module*; Roboter die an derselben Aufgabe arbeiten sollten besser kooperieren, während Roboter die andere Aufgaben ausführen ignoriert und nicht in die Entscheidungsfindung einbezogen werden sollten. Dies vereinfacht einerseits die Entwicklung, da nicht stets das Gesamtsystem betrachtet werden muss, um Änderungen an kleineren Teilaufgaben vorzunehmen und verbessert gleichzeitig die Kooperation, da Roboter innerhalb einer Gruppe sich besser abstimmen können. Die Gruppen werden dabei durch den Role Manager definiert.
- *Optimierung der Zuweisung*; Damit die Roboter als Team spielen, ist es notwendig dass sie gemeinsam spielen und ihre Mitspieler bei den Entscheidungen berücksichtigen. Da prinzipiell alle Roboter hardwaretechnisch gleich aufgebaut sind, kann jeder Roboter jeden Task gleich gut ausführen. Dies ermöglicht auch eine optimierte Zuweisung, bei der zunächst die auszuführenden Tasks ermittelt werden und die tatsächliche Zuweisung dieser zu den Robotern erst im nachfolgenden Schritt erfolgt. Dadurch kann der Gesamtweg des Teams minimiert werden, was in kürzeren Fahrtwegen und somit einer höheren Spieldynamik resultiert.

Die grobe Architekturänderung ist in der nachfolgenden Abbildung gezeigt. Links befindet sich das zuvor genutzte Rule-System, bei der ein komplexes Rule-System die Aufgaben für alle Roboter generiert. Rechts ist der für Bordeaux 23 dargestellte Ansatz, bei der die Roboter Rollen erhalten. Roboter mit derselben Rolle befinden sich in derselben Gruppe und arbeiten gemeinsam an einer Aufgabe, wie der Verteidigung des Strafraums.



Eine weitere Wesentliche Neuerung ist, dass die *Rule-Systeme* zur Steuerung der einzelenen Rollen durch *Zustandsautomaten* ersetzte wurden. Ein wesentlicher Vorteil dieses Schrittes besteht darin, dass einerseits besser kontrolliert werden kann, welche Zustände in welche anderen Zustände übergehen dürfen und andererseits, dass *Hysteresen* verwendet werden können. Diese Schritte stabilisieren das System und verhindern so den ständigen Wechsel von Tasks, falls ein instabiler Zustand (zb. durch Gegnerbewegungen/Rauschen/schlechter Datenqualität/...) eintreten kann, bei denen die Roboter sehr schnell zwischen Zuständen wechseln und daher letztlich keine der Aktionen ausführen.

> **Rule-System**:
> Ein Rule-System ist eine Architektur zur Steuerung von Systemen. Hierbei werden verschiedene Regeln definiert, die an Aktionen gekoppelt sind. Jede Rule besitzt zudem eine Priorität, wobei die Regeln in absteigender Priorität geprüft werden. Ist die Bedingung einer Regel erfüllt, so wird die zugehörige Aktion ausgeführt. Weitere Infos: Wikipedia, Complexica.

> **Zustandsautomat**:
> Ein Zustandsautomat (engl. State Machine) ist eine Architektur zur Steuerung von Systemen. Dabei werden verschiedene Zustände sowie Transitionen zwischen diesen definiert. Das System verbleibt in einem Zustand, bis die Bedingung an einer dort entspringenden Transition erfüllt ist. Jedem Zustand können Aktionen zugeordnet werden, um so ein Verhalten zu steuern. Weitere Infos: Wikipedia, Medium.

> **Hysterese**:
> Eine Hysterese beschreibt ein Prinzip, mit dem sich Zustandsänderungen kontrollieren und stabilieren lassen können. Beispiel: Wird ein Zustand bei

> gewechselt, wenn x > 2 ist, soll dieser erst wieder verlassen werden wenn x < 1 ist. Flackert der Wert zwischen 1,8 und 2,2 so bleibt der Zustand stabil. Weitere Infos: Max Planck Gesellschaft: Hysterese.

Verwendete Rollen und deren Aufgaben

Die genauere Zuweisung der Rollen wird in Role Manager Bordeaux 2023 beschrieben. Es stehen die folgenden Rollen zur Verfügung, deren Details in eigenen Wiki-Einträgen angesehen werden können:

- *Torwart*: Soll das Tor Verteidigen und Ball im Strafraum manipulieren.
- *Verteidigung*: Soll direkte Schüsse auf das Tor verhindern indem am Strafraum gemauert wird. Die Tasks werden hierbei gemäß des Flexwall-Algorithmus bestimmt und zugewiesen.
- *Support*: Soll Pässe verhindern, indem gefährliche Gegner gedeckt werden. Die Funktionsweise des Algorithmus findet sich hier.
- *Offensive*: Soll den Ball erobern, sich für Pässe anbieten sowie Torschüsse ermöglichen. Dies wird ausführlicher hier beschrieben.

Support

Der Support stellt die aktive Verteidigung dar. Der Support wurde für Bordeaux im Rahmen des Taskmanager_Bod23 entwickelt und dort erstmalig eingesetzt. Das Hauptziel des Supports besteht in der Manndeckung, durch die Pässe verhindert und abgefangen werden sollen. Neben dem Support besteht die Verteidigung auch noch aus der Flexwall, die den Strafraum vor direkten Torschüssen schützt, falls ein Pass nicht abgefangen werden kann.

> Da sich die Strategie schnell und stetig ändert, ist dieser Artikel als ein Einstieg in die Thematik zu verstehen und kann keinen Anspruch auf Aktualität erheben. Hierzu ist der tatsächliche Code heranzuziehen.

Grundprinzip

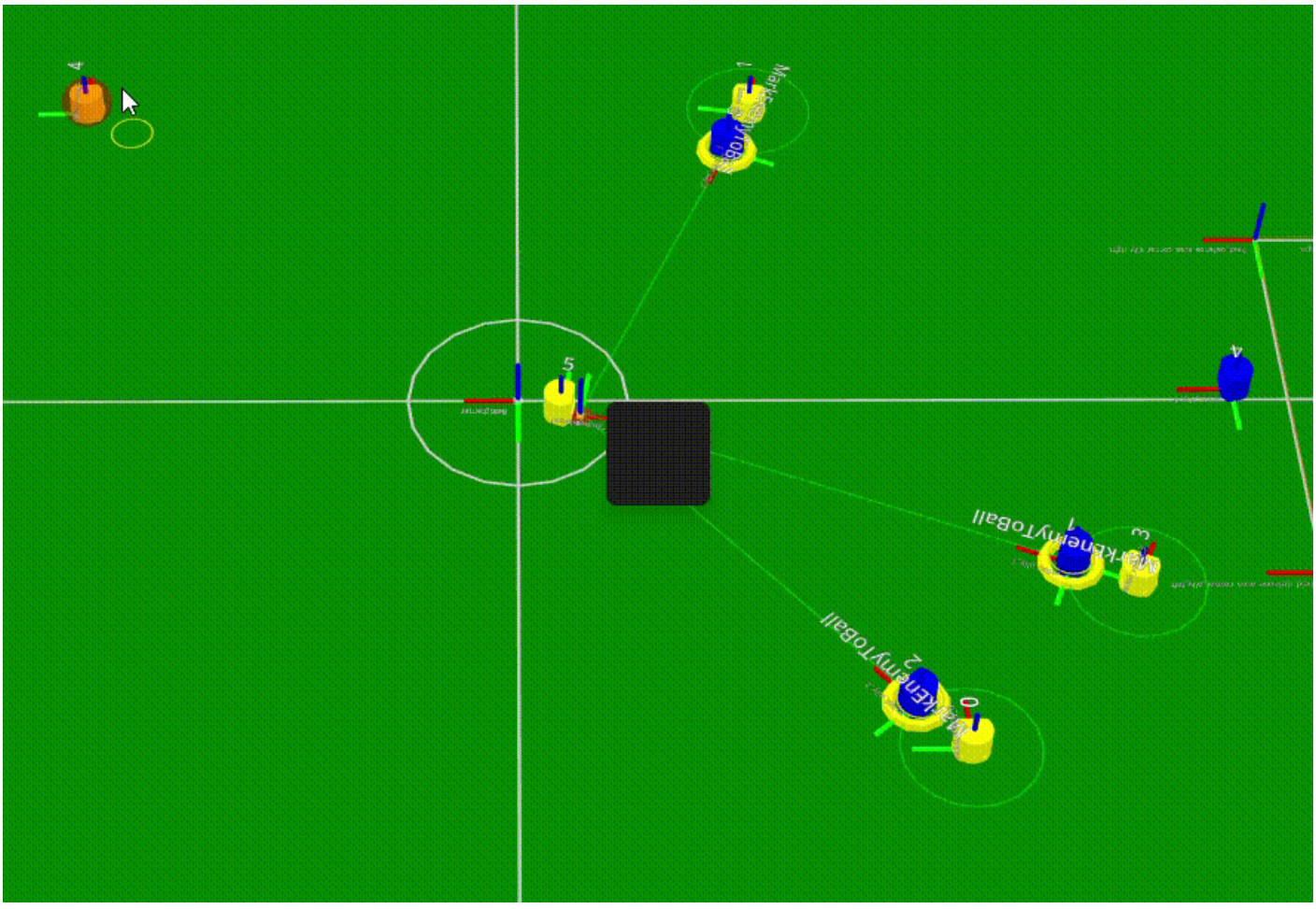
Das Ziel ist die Manndeckung, bei der verhindert werden soll, dass Pässe zu gefährlichen Gegnern gespielt werden können. Hierzu positionieren sich die Roboter nahe des zu deckenden Spielers auf die Sichtlinie vom Ball zu diesem Spieler.
Um die Entscheidung zu treffen, welche Gegner gedeckt werden sollen, müssen zunächst die Gefahrenlevel (=Threat) der Gegner eingeschätzt werden. Hierzu stellt der Observer Funktionen zur Verfügung, die im Allgmeinen auf der Nähe des Roboters zum Tor und zum Ball basieren. Ausnahmen stellen der Torwart und der Ballführende Spieler dar, die nicht berücksichtigt werden sollten, da eine Deckung dieser im Allgemeinen nicht sinnvoll ist.

1. Die verbleibenden Gegner werden nach absteigendem Threat geordnet. Im Allgmeinen kann davon ausgegangen werden, dass die Anzahl der Supporter geringer als die der Gegner ist, weshalb nicht alle Gegner gedeckt werden können. Stattdessen werden nur die Gegner mit hohem Threat betrachtet, der Rest verworfen.
2. Es wird für jeden zu verteidigenden Gegner die Zielposition geschätzt; Diese liegt etwa bei dem Gegner, mit einem Offset in Richtung Ball. Dieser Schritt sollte in zukünftigen Softwareversionen direkt von der Pfadplanung übernommen werden.
3. Es wird eine Kostenmatrix aufgestellt, die die Distanzen von jedem Supporter zu jeder Zielposition beinhaltet. Diese ist die Basis für den Munkres Algorithmus, mit dem eine ideale Zuordnung der Roboter zu den Zielen gefunden wird. Hier hat sich als Kostenmaß die euklidische Distanz bewährt, da Roboter im Allgemeinen versuchen, bei ihrem aktuellen Ziel zu verbleiben, auch wenn sich die zu deckenden Gegner ändern.
4. Die Roboter werden gemäß der Ergebnisse des Munkres-Algorithmus auf die verschiedenen Gegner verteilt.

Durch diesen Vorgang werden die *n* gefährlichsten Gegner von *n* Supportern gedeckt. Die Deckung ist dabei jedoch nur als eine Vorpositionierung zu verstehen. Sollte der Gegner dennoch einen Pass-Versuch unternehmen, so wird dieser oft in den Lauf erfolgen, wodurch an dem Supporter vorbeigespielt wird. Daher sollten die Supporter Intercept-Mannöver durchführen, sobald ein Schuss in Richtung des Roboters detektiert wird, um den Ball aktiv abzufangen.

Support in der Praxis

Im folgenden Video wird die Funktion des Supports in einem Simulationsszenario demonstriert:



Erfahrungen und Kritik

- Das Verfahren hat sich in der Vergangenheit allgemein als sinnvoll erwiesen. Das Abfangen von Pässen ist absolut kritisch, da so Torschüsse frühzeitig verhindert werden. Kann der Gegner ungestört passen, so gibt ihm das die Freiheit Lücken in der Verteidigung zu erspielen und auszunutzen. Der Support behindert dabei die Freiheiten des Gegners.

- Da der Suppport im Spielfeld und meist fern vom Strafraum agiert, können die Spieler auch schnell in die Offensive wechseln, wenn ein Ball erfolgreich abgefangen wird, um so einen Konter durchzuführen.
- Der Support versucht stets die aktuelle Position des Gegners zu decken. Dies berücksichtigt dessen Fahrt nicht und führt dazu, dass wir leicht hinterherhinken.
- Wenn sich Gegner "verklumpen", so führt dies dazu, dass ggfs. einige Support versuchen die Gegner zu decken und dabei die Klumpenbildung verstärken. Dies ist nicht sinnvoll, da sich die Roboter dabei gegenseitig behindern und dies keinen signifikanten Mehrwert hat.
- Es hat sich als sinnvoll erwiesen, neben Torwart und Ballführer auch die gegnerischen Verteidiger auszuschließen, bzw. die Threat-Berechnung so zu gestalten, dass diese nicht berücksichtigt werden. Hier sollten alle Spieler ausgeschlossen werden, die sich im oder um den gegnerischen Strafraum befinden.

Weiterentwicklungen

Aktuell bleibt das Konzept für Eindhoven unverändert.

Ideen für zukünftige Weiterentwicklungen

- *Berücksichtigung der Gegnergeschwindigkeit*: Es sollte abgeschätzt werden, wo der Gegner in den nächsten x ms sein wird und diese Position gedeckt werden. Dies sollte das Abfangen von Pässen in den Lauf verbessern.
- *Verbesserung des Threat-Levels*: Gegner die sehr nah am Ball sind, müssen ggfs. nicht gedeckt werden, da sie keine große zusätzliche Gefahr darstellen.
- *Verbesserung des Intercept-Verhaltens*: Der ideale Intercept Punkt sollte sinnvoller bestimmt werden, sodass der Roboter schnellstmöglich den Ball erhält. Außerdem wäre es sinnvoll, den Punkt so zu wählen, dass er weit weg von Gegnern ist, damit diese den folgenden Konter nicht so einfach behindern können.
- *Clusterverteidigung*: ähnlich wie beim Flexwall-Algorithmus können ggfs. Gegner zusammengefasst und gemeinsam verteidigt werden. Dies reduziert die Anzahl an notwendigen Support-Spielern.

Offensive

Die Offensive ist Teil des Taskmanager. In diesem Artikel wird vor allem auf den in im Rahmen des Taskmanager_Bod23 entwickelten Algorithmus eingegangen und das zugrundeliegende Konzept vorgestellt. Außerdem werden die Erfahrungen sowie Ideen zur Weiterentwicklung dokumentiert.

> Da sich die Strategie schnell und stetig ändert, ist dieser Artikel als ein Einstieg in die Thematik zu verstehen und kann keinen Anspruch auf Aktualität erheben. Hierzu ist der tatsächliche Code heranzuziehen.

Die Positionierung von offensiven Spielern war das Kernthema unseres eTDP 2024 (siehe Anhang links).

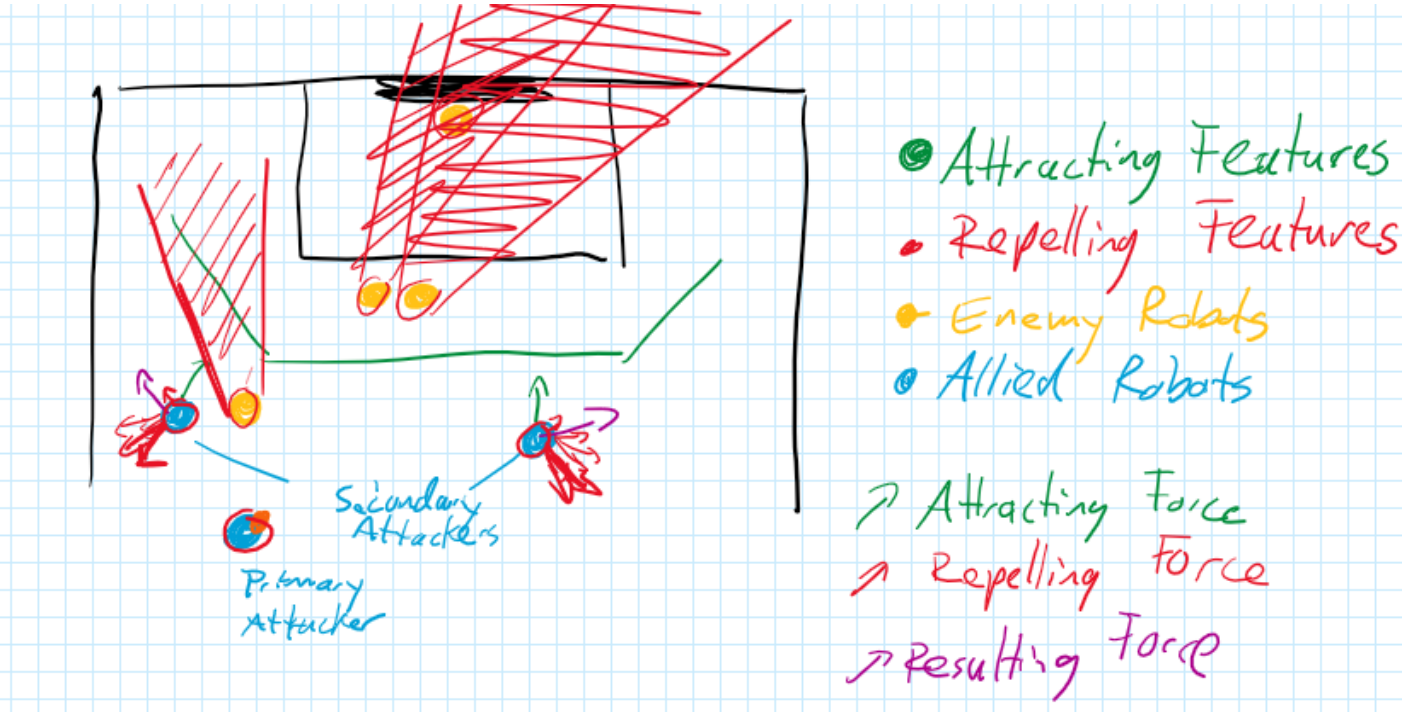
Motivation und Hintergrund

Das Ziel der Offensive besteht vor allem in der Eroberung und Kontrolle des Balles, sowie den Vorbereitungen für einen erfolgreichen Torschuss. Da stets davon ausgegangen werden kann, dass der Gegner versuchen wird, den eigenen Torschuss zu verhindern, müssen die Bewegungen und Möglichkeiten der gegnerischen Roboter zu jedem Zeitpunkt berücksichtigt werden.
Anders als die Verteidigung kann eine Offensive nicht rein reaktiv funktionieren. Eine rein reaktive Offensive kann schon an einer sehr simplen Verteidigung scheitern. Die aktive Vorausplanung erfordert jedoch deutlich komplexere und umfangreiche Algorithmen, bei denen die Verteidiger aktiv berücksichtigt und umgangen werden. Hier können Konzepte aus der Spieltheorie angewandt und herangezogen werden, wie beispielsweise Spielbäume, auf die im Weiteren auch eingegangen werden wird.

Offensive Bangkok 2022

Die erste Iteration der Offensive wurde im Rahmen des Taskmanager Bangkok 2022 entwickelt und getestet. Damals wurde ein zentrales Rule-System für die gesamte Strategie verwendet, was den Code schwer anpassbar gemacht hat und sich auch unmittelbar auf die Verteidigung etc. ausgewirkt hat.
In diesem Rule-System waren verschiedene Offensive Rules, die nach ihrer Priorität abgearbeitet wurden:

1. Torschuss: Wenn ein Roboter den Ball hat und die Torwahrscheinlichkeit hoch ist, soll ein Torschuss unternommen werden
2. Passen: Wenn kein Torschuss ist und ein Pass durchgeführt werden kann, soll ein Pass ausgeführt werden
3. Mit Ball fahren: Sind weder Torschuss noch Pass möglich, soll der Roboter mit Ball fahren
4. Emergency Shot: Sind weder Torschuss noch Pass noch Dribbling möglich (z.B. weil die maximale Fahrdistanz mit Ball erreicht wurde), wird der Ball weggeschossen
5. Get Ball: Ist unser Team aktuell nicht im Ballbesitz, soll der nächste Roboter den Ball holen/stehlen
6. Offensive Movement: Roboter (die nicht den Ball haben) sollen sich freilaufen, wobei sie von dem gegnerischen Strafraum angezogen und von Gegnern sowie deren "Schatten" ausgehend vom Ball abgestoßen werden. Dies wird in der nachfolgenden Abbildung gezeigt:

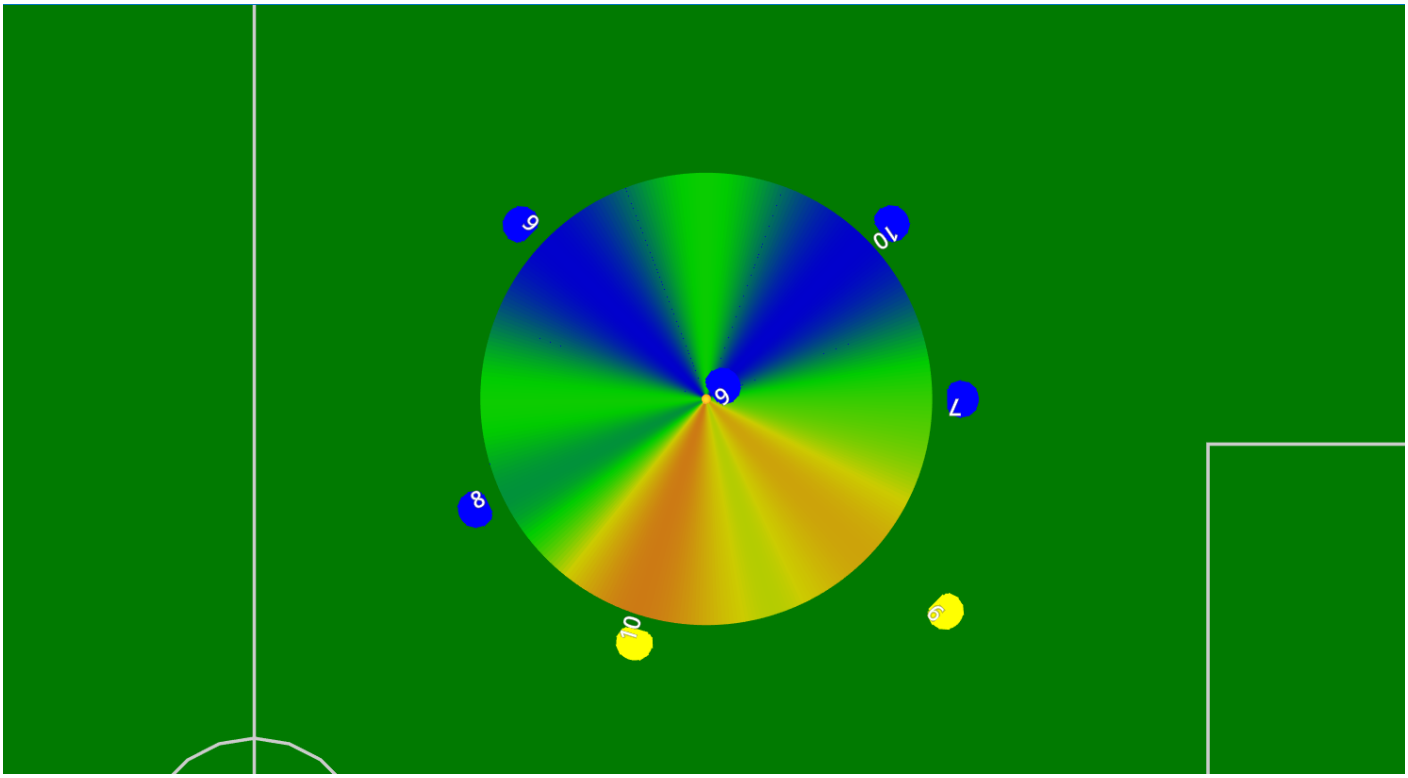


Dieses System stellt eine reine Reaktion auf den Ist-Zustand dar. Gegner werden zwar in der Entscheidung ob ein Schuss durchgeführt werden soll berücksichtigt, ebenso bei der Positionierung im Offensiven Movement. Allerdings sind dies passive Entscheidungen, die das Verhalten und die Möglichkeiten der Gegnern nicht direkt berücksichtigen und um diese herumplanen. Außerdem hat sich gezeigt, dass das Offensive Movement oftmals in nicht idealen Positionen resultiert hat, da die Roboter zu weit weg waren, wenn der Ball in unserer eigenen Hälfte war, außerdem fahren die Roboter zwar von Gegnern weg, jedoch nicht zwingend auf eine Weise, die sie besser anspielbar macht.

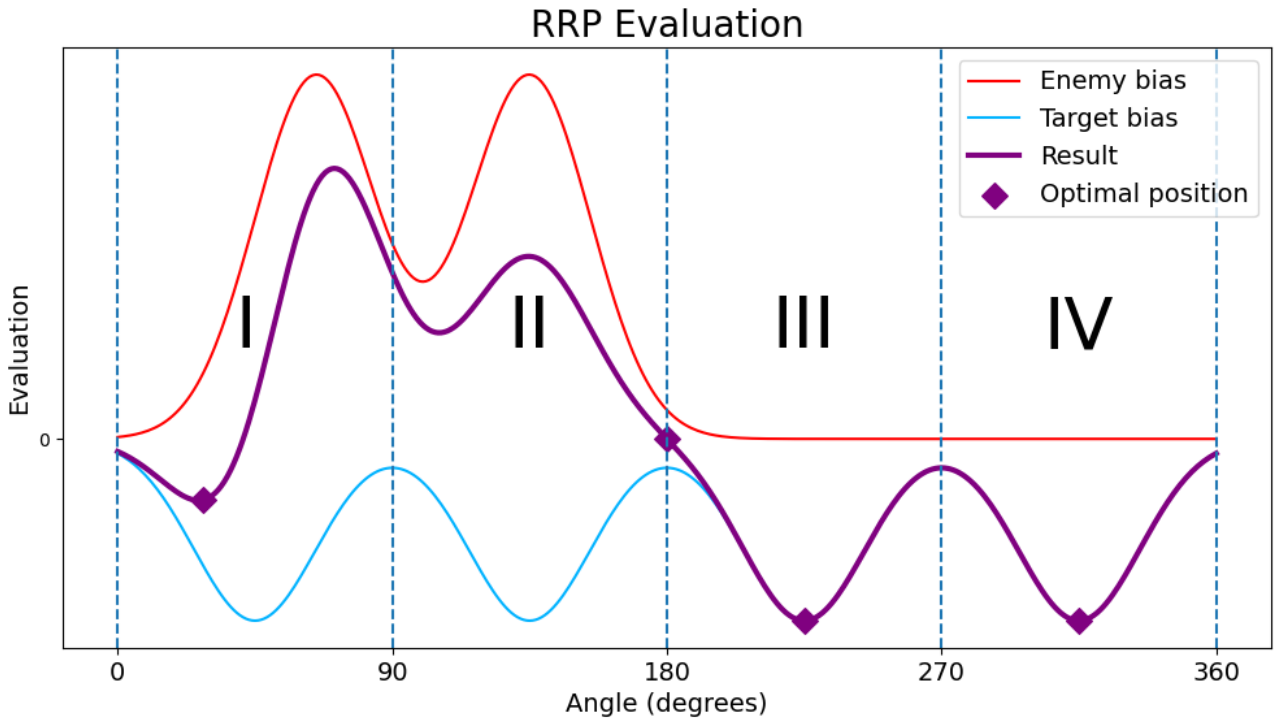
Offensive Crailsheim 2023

Vor dem RoboCup in Bangkok wurde ein Testtunier in Crailsheim gespielt, auf dem erstmalig die zweite Iteration der Offensive getestet wurde. Das wesentliche Rule-System aus Bangkok wurde

von der Logik übernommen, allerdings mit dem Rollensystem, welches durch den Rolemanager_bod23 vorgegeben wird. Dadurch war der Code einfacher und verständlicher, da die Offensive von der Defensive entkoppelt worden konnte. Außerdem wurde auch das offensive Movement überarbeitet.
Anstatt vom Strafraum angezogen zu werden und dabei ggfs. zu weit vom Ball wegzufahren, wurde der Abstand der Roboter auf einen vorgegebenen Abstand festgesetzt. Offensive Spieler konnten sich somit nur noch auf einer Kreisbahn mit etwa zwei Meter Radius um den Ball herum bewegen. Um Gegnern auszuweichen und Soll-Positionen zu definieren, wurden dabei Punkte definiert, von denen die Roboter angezogen bzw. abgestoßen werden.



Das Prinzip folgt einer Optimasuche auf einem Graphen. Der Graph beschreibt dabei die Güte einer Position bezogen auf den Winkel relativ zum Ball. In dieser Überlegung werden Gegner durch eine positive Gausskurve repräsentiert, wodurch im Graphen ein "Berg" an der Gegnerposition entsteht. Die Sollpositionen werden durch negative Gausskurven repräsentiert, die zu "Tälern" führen. Das Ziel der offensiven Robotern ist es, zum Minimum in dem ihnen zugewiesenen Quadranten zu fahren. Dies soll verhindern, dass mehrere Roboter dieselbe Position anfahren sollen. Welche Quadranten verwendet werden sollen, hängt von der Ballposition ab. im Allgemeinen werden dabei Quadranten besetzt, die nahe der Spielfeldmitte sind.



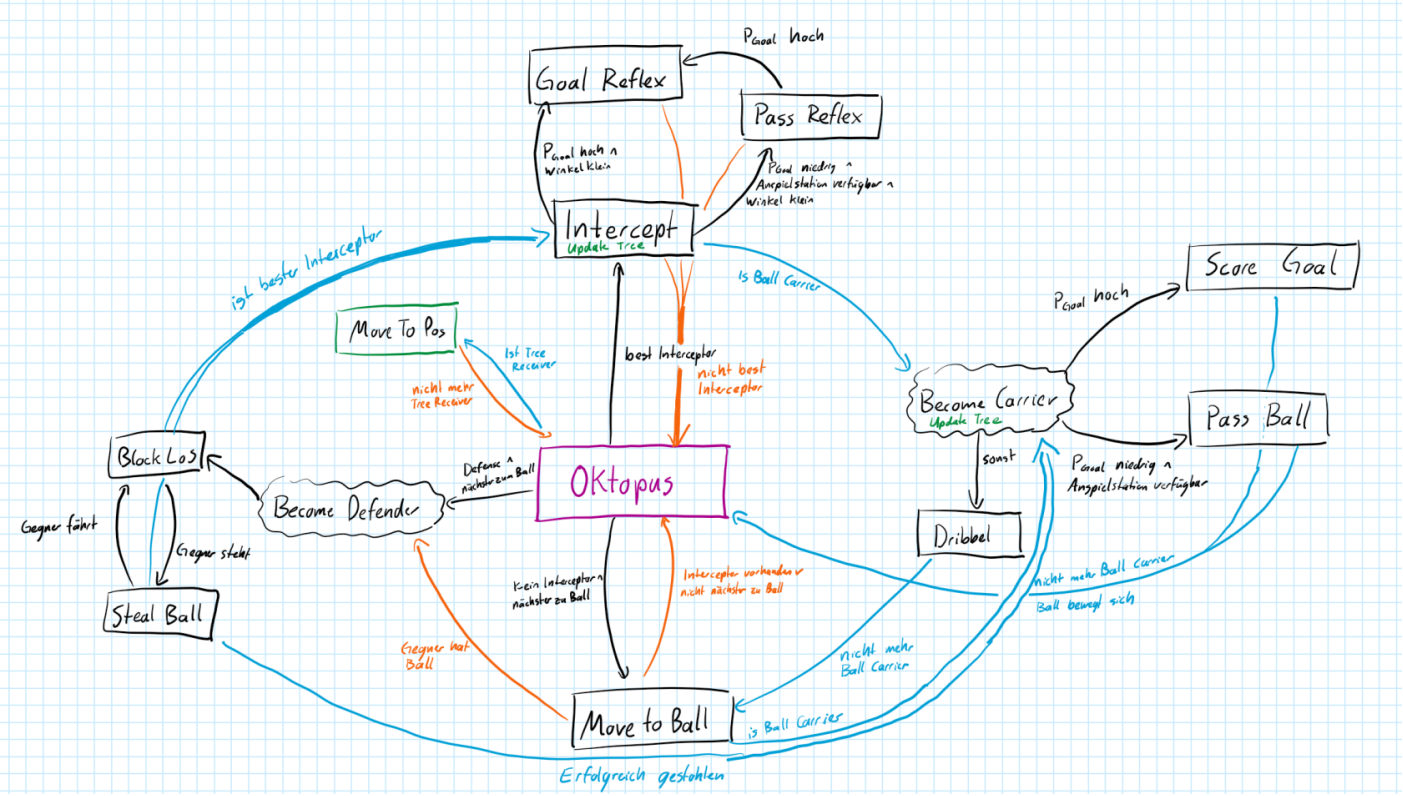
Die Spiele in Crailsheim haben gezeigt, dass es mit diesem Ansatz möglich ist, die Entfernung vom Ball zu stark zu limitieren, dass die Roboter stets anspielbar sind. Außerdem fahren die Roboter nicht nur vor Gegnern weg, sondern weichen dabei auch in eine Richtung aus, die die Anspielbarkeit erhöht.
Auch wenn dieser Ansatz eine Verbesserung darstellt, ist es trotzdem keine hinreichend gute Lösung. Das Hauptproblem dieses Lösungsansatzes ist, dass das System durch die Begrenzung auf den Kreis sehr starr ist. Desweiteren weichen die Roboter zwar Gegnern aus, versuchen dabei aber nicht aktiv eine Gefahr darzustellen. Selbst wenn ein Pass erfolgreich ausgeführt wird, ist dies keine Garantie, dass dadurch ein nennenswerter Fortschritt bezüglich der Torwahrscheinlichkeit entsteht. Die Folge dessen ist der Verwurf dieses Ansatz zu Gunsten einer komplexeren, aktiven Offensive.

Aktive Offensive: Bordeaux 2023

Wie aus dem vorherigen Abschnitt hervorgeht, ist der frühere, passive Ansatz nicht in der Lage, eine wirkliche Bedrohung für den Gegner darzustellen. Selbst schwache Defensiven können nicht aktiv umgangen werden. Daher wurde vor dem RoboCup in Bordeaux ein besonderer Fokus auf die Entwicklung einer aktiven Offensive gesetzt, deren Konzepte im folgenden Erläutert werden.

Grundstruktur

Eine wesentliche Änderung war der in Taskmanager_Bod23 beschriebene Wechsel von einem Rule-System zu einem Zustandsautomaten. Dieser Zustandsautomat steuert alle Spieler mit der offensiven Rolle, wobei die verschiedenen Roboter in unterschiedlichen Zuständen sein können. Generell ist stets ein Zustand aktiv, der sich um die Ballmanipulation (Ball holen, Ball abfangen, schießen, dribbeln...) kümmert und nur von genau einem Roboter ausgeführt werden darf. Diese Beschränkung ist wichtig, da sich Roboter ansonsten gegenseitig den Ball wegnehmen oder behindern würden. Die restlichen Roboter werden verwendet, um sich freizulaufen und für Pässe anzubieten. Der grobe Aufbau des Zustandsautomaten befindet sich in der folgenden Grafik, wobei für die Einfachheit nicht alle Transitionen und Bedingungen eingezeichnet wurden. Die verschiedenen Zustände werden im Folgenden genauer erläutert.



Oktopus

Dies ist der Standardzustand. Roboter, die noch keinem Zustand zugewiesen werden starten hier. Bei Oktopus handelt es sich um einen Algorithmus, bei dem sich die Roboter freilaufen, um sich so als potenzielle Anspielstation anzubieten. Sobald der Roboter sich in einer guten Position befindet und dadurch in eine Passkette eingeplant wird, wechselt er zu *MoveToPos*. Ist das nicht der Fall, sollte der Roboter durch Oktopus weiterhin versuchen, die eigene Position zu verbessern, bis er als Anspielstation herangezogen wird. Die Funktionsweise des Algorithmus wird in einem eigenen Artikel zu Oktopus erläutert.

Primary Attacker

Wie bereits erwähnt, sollte stets genau ein Spieler mit der Ballführung beauftragt werden, dieser wird im Folgenden als Primary Attacker bezeichnet. Dies ist im Allgemeinen der Roboter, welcher am nächsten zum Ball ist. Hat dieser aktuell noch nicht den Ball in der Lichtschranke, sollte er *MoveToBall* bzw. *StealBall* Ausführen, je nachdem ob grade ein Gegner bereits den Ball hat oder nicht. Wenn sich der Gegner bewegt, sollte statt *StealBall* eher *BlockLoS* (LoS = LineOfSight) ausgeführt werden, da in diesem Fall das Stehlen des Balles schwierig ist und der Gegner so an einem Pass oder Schuss gehindert werden kann. Sobald der Gegner mit dem Ball zum Stillstand kommt, kann ein Balldiebstahl versucht werden.
Diese Fälle sind sinnvoll für einen sich nicht oder nur langsam bewegenden Ball. Wurde der Ball soeben geschossen, sollte stattdessen versucht werden, dass der Primary Attacker diesen Ball abfängt. Da sich der Ball schnell bewegt, sollte auch nicht der nächste Roboter als Primary Attacker verwendet werden, sondern stattdessen der Roboter, der sich am nächsten zur Balltrajektorie befindet, was durch den *Best Interceptor* aus dem observer vorgegeben wird. Anstatt eines Intercept-Manövers kann auch direkt ein Reflexschuss durchgeführt werden, bei dem der Ball nicht angenommen, sondern direkt weitergeschossen wird, sofern der Winkel dabei klein genug ist und es ein sinnvolles Ziel gibt. Für das Ziel kann einerseits das Gegnertor verwendet werden, sofern die Torwahrscheinlichkeit über dem von Adathresh festgelegtem Schwellwert liegt. Gleichermaßen kann auch ein Reflexpass ausgeführt werden, wobei diese Funktion während des Wettkampfes in Bordeaux aufgrund der Anfälligkeit für Software-Bugs entfernt werden musste.
Pässe und Torschüsse können auch normal ausgeführt werden, wenn der Roboter im Ballbesitz ist. Dabei hat ein Torschuss stets Priorität. Ob ein Torschuss ausgeführt werden sollte, hängt wie schon beim Reflexschuss davon ab, ob die aktuelle Torwahrscheinlichkeit über dem von Adathresh festgelegten Schwellwert liegt. Ist dies der Fall, wird ein Schuss auf den besten Torpunkt ausgeführt, dessen Berechnung ebenso wie die der Torwahrscheinlichkeit hier eingesehen werden kann.
Ist ein Torschuss nicht möglich, wird stattdessen ein Passbaum aufgebaut, der verschiedene Passketten evaluiert. Der Passbaum beinhaltet mögliche Pässe zu jedem anderen offensiven Mitspieler, wobei auch Pässe in den Lauf berücksichtigt werden. Die Pässe werden hinsichtlich der Wahrscheinlichkeit, dass der Ball nicht von einem Gegner abgefangen wird, sowie der Torwahrscheinlichkeit des Empfängers nach Erhalt des Balles evaluiert. Dabei wird auch berücksichtigt, dass der Ball vom Empfänger zu einem weiteren Roboter gepasst werden könnte, wodurch Passketten, Doppelpässe etc. entstehen. Aus dem Baum wird dann die Passkette ausgesucht und gespielt, die einerseits die geringste Abfangwahrscheinlichkeit durch Gegner zulässt und andererseits die resultierende Erfolgschance auf ein Tor maximiert.
Sollten alle Mitspieler gedeckt oder aus anderen Gründen nicht verfühgbar sein, wird als Notlösung ein Dribbling durchgeführt. Das Dribbling besteht dabei aus schwachen Schüssen in Richtung des Gegnerischen Tores, wobei hierbei auch Gegnern ausgewichen wird. Durch das Dribbling sollen die Mitspieler Zeit gewinnen, um sich freilaufen und als Anspielstation anbieten zu können und gleichzeitig ein Fortschritt mit dem Ball zum gegnerischen Tor erzielt werden.

MoveToPos

Dieser Zustand wird von offensiven Spielern eingenommen, die in der aktuellen Passplanung durch den Passbaum als Anspielstation eingeplant sind. In diesem Zustand fahren die Roboter an die Position, an der später der Ball angenommen werden soll. Wird der Roboter in der aktuellen Passplanung nicht berücksichtigt oder ändern sich die Passpläne, sodass er nicht länger berücksichtigt wird, so wechselt der Roboter in den Zustand Oktopus, um sich solange weiter frei zu laufen, bis er erneut in eine Passplanung miteinbezogen wird.

Erfahrungen und Evaluation

Die Offensive wurde erstmalig in Bordeaux 2023 getestet und hat sich schnell als eine deutliche Verbesserung zum vorherigen Stand aus Crailsheim herausgestellt. Durch den Aufbau der Passketten in den Passbäumen war es unserem Team möglich, sehr effizient Pässe zu spielen und so um die gegnerische Verteidigung herumzuspielen. Während die Crailsheimer Offensive zwar in der Lage war, Namec mit stillstehenden Robotern innerhalb von knapp 10 Minuten mit 10:0 zu besiegen, konnte die Bordeaux-Offensive dies in der Hälfte der Zeit schaffen und auch gegen Teams, die leichten Widerstand geleistet haben, indem beispielsweise Mauern am Strafraum gebildet wurden. Das Verfahren an sich hat sich somit als sinnvoll erwiesen, trotzdem sich auch Schwachstellen und Verbesserungsmöglichkeiten aufgefallen, die im folgenden aufgelistet sind:

- *schlechte Dynamik*: Die Roboter benötigen im Allgemeinen zu lange, um sich mit dem Ball zu drehen. Dies ist in erster Linie ein Problem der Hardware und des Plannings, sollte aber in der Planung berücksichtigt werden, da Passketten sonst zu früh unterbrochen werden.
- *schwaches Zweikampfverhalten*: Gegen aggressive Roboter ist es uns allgemein schwer gefallen, Pässe zu spielen und Fortschritt zu erreichen. Bereits ein einzelner Roboter, kann durch eine LineOfSight-Verteidigung sämtliche Passversuche unterbinden. An dieser Stelle wäre einerseits ein Lupfer sinnvoll, um über die Gegner rüber schießen zu können, alternativ wären auch Konzepte denkbar, bei denen andere Offensive Spieler die Verteidiger blockieren, sodass der Primary Attacker einen Schuss ausführen kann. Das Zweikampfverhalten muss jedoch auch in der Defensive verbessert werden; Selbst wenn es uns gelingt, dem Gegner den Ball abzunehmen, sind meist keine sinnvollen Möglichkeiten vorhanden, bevor der Gegner selbst in den Zweikampf geht und den Ball zurückerobert. Auch hier wären Lupfer und unterstützenden offensive Spieler von Vorteil.
- *Lange Rechenzeiten*: Durch die Vielzahl an Möglichkeiten, mit denen Pässe potentiell durchgeführt werden könnten, ist die damit verbundene Rechenzeit sehr hoch. Dies wurde zwar auf unter eine Sekunde reduziert, jedoch auf Kosten der Suchtiefe und -breite. Hier wäre eine effizientere Berechnung sowie evtl. eine Auslagerung der Berechnung auf die Grafikkarte sinnvoll. Andere Teams wie Zjunlict wenden dies bereits an und beschreiben den Vorgang in einem ihrer ETDPs: ZJUNlict 2020.
- *schwache Ballannahme*: Bei den Pässen ist der Ball oft vom Roboter abgeprallt. Dies ist einerseits auf die hardwareseitige Dämpfung zurückzuführen, andererseits sollten jedoch auch die Skills angepasst werden, sodass der Roboter z.B. durch eine bessere Einstellung der Dribbling-rolle und ggfs. durch eine Fahrt zurück den Ball besser annimmt.

- *Instabiler Zustand beim Primären Angreifer* Sind Roboter ähnlich nah am Ball, kann die Zuweisung des Primären Attacker teils mehrmals pro Sekunde wechseln, sodass keine Roboter letztlich an den Ball heranfährt. Hier könnte eine Hystere helfen.

Weiterentwicklungen für Eindhoven

Für Eindhoven soll das Konzept weiterverwendet und weiterentwickelt werden. Diese Änderungen beinhalten:

- Ausgliederung des primary Attacker als eigene Rolle; Dies soll Visualisierung verbessern und den Zustand stabilisieren. Außerdem vereinfacht dies den Zustandsautomat erheblich, da weniger Zustände benötigt werden.
- Bessere und Effizientere Berechnung der möglichen Passpunkte
- Eingliederung in das Eventsystem; Hierdurch sollen Berechnungen seltener und nur wenn sie tatsächlich gebraucht werden durchgeführt werden, um so die allgemeine Performanz zu steigern.

Ideen für Weiterentwicklungen

- *Just in Time Pässe*: Um den Gegner über das Ziel des Passes in Unklaren zu lassen, sollte der Empfänger so zur Position fahren, dass er wenige Millisekunden vor dem Ball dort eintrifft. Möglicherweise könnte er auch erst in die entgegen gesetzte Richtung fahren, um Gegner zusätzlich zu verwirren. Hierbei ist jedoch zu beachten, dass je nachdem wie viele Hindernisse sich auf dem Weg zur Passposition befinden, der Weg ggfs. etwas länger dauern kann.
- *Bessere Prädiktion des Gegnerverhaltens*, Um besser um den Gegner herum zu spielen, wäre es gut, dessen verhaltensweisen besser vorhersehen zu können. Vorsicht jedoch vor Overfitting, der Gegner wird auf unsere veränderte Spielweise ggfs. reagieren.