Lastenheft für *Fantastic Feasts*

Version 2018-12-19

Changelog

- **2018-12-20:** Schnatzverhalten bei Überlänge präzisiert (ignoriert Effekte wie schnappende Niffler, etc.). Wiedereinsetzen von verbannten Spielern präzisiert.
- **2018-12-19:** Rundenphasen für Quidditch-Spieler mit Beispiel verdeutlicht.
- **2018-12-13:** Bedingung für Disqualifikation nach Fouls vervollständigt. + Einige Stellen sprachlich verbessert.
- **2018-12-06:** Verschiedene Details präziser formuliert.
- **2018-11-28:** Regeln für Fanblock präzisiert.

Abstract

Dieses Lastenheft beschreibt die Anforderungen an das Spiel *Fantastic Feasts*, welches im Softwaregrundprojekt entwickelt werden soll. Dieses Dokument wird wahrscheinlich zu späteren Zeiten überarbeitet oder erweitert werden, um geänderte oder präzisierte Anforderungen widerzuspiegeln, die sich im Lauf des Projekts ergeben können.

Kapitel 1 gibt eine Übersicht zur Architektur und den Komponenten des zu entwickelnden Produkts, und legt einige technische Randbedingungen fest.

Kapitel 2 beschreibt die Spielregeln des eigentlichen Spiels *Fantastic Feasts*.

Kapitel 3 listet verbindliche und optionale Anforderungen an die einzelnen Komponenten des verteilten Systems auf.

Kapitel 4 macht Vorgaben zum Ablauf des Projekts und beschreibt den einzuhaltenden Entwicklungsprozesses.

Kapitel 5 legt die Abnahmekriterien für das Produkt fest.

Inhaltsverzeichnis

1 Produkt und Einsatzszenarien

Im Softwaregrundprojekt soll die **rundenbasierte Quidditch-Simulation *Fantastic Feasts*** als Multiplayerspiel entwickelt werden. Dieses Kapitel beschreibt die Architektur des verteilten Systems und listet seine Komponenten auf. Danach werden technische Vorgaben gemacht.

1.1 Komponenten und Architektur

Das zu entwickelnde Produkt ist eine verteilte Anwendung mit **Client-Server-Architektur.** Die Clients kommunizieren dabei nur mit dem Server, aber nicht untereinander. Das System besteht aus folgenden Komponenten (vgl. Abb. 1.1):

- Ein **Server**, der eine Partie *Fantastic Feasts* zwischen zwei Clients verwalten und abwickeln kann. Er beinhaltet die Spiellogik. Einstellungen für eine konkrete Partie werden über eine **Partie-Konfiguration** vorgegeben. Die Partie-Konfiguration bestimmt z.B., wieviel Zeit den Clients für bestimmte Aktionen gewährt wird, wie die Wahrscheinlichkeiten von Ereignissen im Spiel sind, und wieviele Runden vergehen dürfen, bis die Bedingung für überlange Partien eintritt.
- Ein **Benutzer-Client** mit graphischer Oberfläche, der es einem einzelnen menschlichen Benutzer ermöglicht, entweder aktiv als Mitspieler an einer Partie teilzunehmen, oder als passiver Zuschauer im Zuschauermodus eine Partie zwischen zwei anderen Clients zu verfolgen.
- Ein **KI-Client** um auch vereinsamten Spielern ein spannendes Multiplayer-Erlebnis bieten zu können. KI-Clients werden von einer Künstlichen Intelligenz gesteuert und können wie menschliche Mitspieler an einer Partie teilnehmen.
- Einen **Quidditchteam-Editor,** mit dem man sich ein Quidditchteam für das Spiel zusammenstellen kann. Ein Quidditchteam besteht aus den Charakteren der Spieler und ihrer Ausrüstung, sowie den magischen Fans. Der Client übergibt vor Beginn einer Partie eine vom Editor erstellte **Quidditchteam-Konfiguration** an den Server. Der Editor erlaubt auch das Erstellen und Editieren von **Partie-Konfigurationen.**

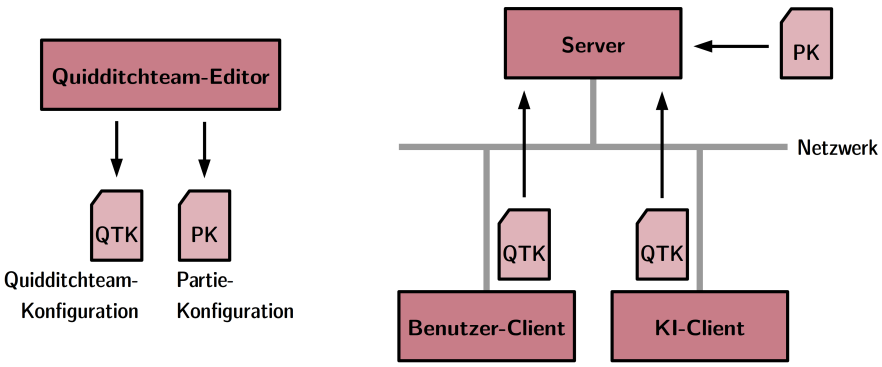

Abbildung 1: Komponenten und Architektur des Multiplayerspiels.

Jedes Team muss am Ende der Projektlaufzeit ein funktionierendes Spiel mit allen Komponenten abliefern. Sie müssen dabei die Komponenten **Benutzer-Client** und **KI-Client** selbst entwickeln. Von den beiden Komponenten **Server** und **Quidditchteam-Editor** müssen Sie nach eigener Wahl nur *eine* implementieren und bekommen die jeweils andere von einem anderen Team auf einer Messe zur Verfügung gestellt (s. Abschnitt 4.10).

1.2 Anwendungssprache, Implementierungssprache und Dokumentationssprache

Um den Benutzern und Zuschauern das Verständnis der Vorgänge während eines Quidditch-Spiels zu erleichtern (vgl. [6]) soll die **Benutzerschnittstelle** der Anwendung (d.h. aller Komponenten) in Deutsch gestaltet werden. Die **Implementierungssprache** (Bezeichner im Source Code und Kommentare) soll jedoch Englisch sein.

Sonstige **Dokumente** wie Benutzerhandbuch, Testdokumentationen, Projekttagebuch, usw. können auf Deutsch oder Englisch verfasst werden.

1.3 Programmiersprachen und Technologien

Als Programmiersprachen empfehlen wir **Java, JavaScript, C# und C++** [1]. Verschiedene Komponenten können dabei auch in unterschiedlichen Sprachen entwickelt werden. Jedes Team kann selbst entscheiden, welche Sprachen, Bibliotheken, Frameworks und Technologien es verwendet. Dies setzt allerdings voraus, dass innerhalb des Teams ein Konsens über diese Auswahl besteht, und der Tutor als Vertreter des Auftraggebers zustimmt. Da alle Teammitglieder an der Implementierung mitwirken sollen, sollten Sprachen gewählt werden, mit denen alle hinreichend vertraut sind.

1.4 Plattformen

Die Komponenten **Benutzer-Client** und **Quidditchteam-Editor** müssen jeweils auf mindestens einer der folgenden Plattformen laufen:

- eine der verbreiteten, aktuellen Linux-Distributionen
- eine aktuelle Version von Microsoft Windows
- basierend auf Webtechnologien, in einem aktuellen, standardkonformen Browser

Die Komponenten **KI-Client** und **Server** müssen inklusive aller Abhängigkeiten als Docker-Container lauffähig sein (s. Abschnitt 4.5).

1.5 Netzwerkkommunikation und Nachrichtenprotokoll

Als Netzwerkprotokoll für die Kommunikation der Komponenten untereinander ist das **WebSocket-Protokoll** [4] vorgegeben. Über die WebSocket-Verbindung werden **Textstrings im UTF-8 Encoding** ausgetauscht, die im Format **JSON** [5] repräsentierte Nachrichten des anwendungsspezifischen Spielprotokolls enthalten. Das Spielprotokoll wird vom Standardisierungskomitee definiert (s. Abschnitt 4.9).

1.6 Formate für Konfigurationsdateien

Quidditchteam-Konfigurationen und **Partie-Konfigurationen** sollen im Format **JSON** repräsentiert werden. Die genauen Schemata definiert das Standardisierungskomitee.

[1] Die hauptsächliche Lehrsprache in Grundlagenveranstaltungen ist Java. Jeder im Team sollte also zumindest Grundkenntnisse in Java haben. Teams können auch andere Sprachen verwenden (z.B. ihre KI in Haskell schreiben), aber kein Teammitglied soll von der Arbeit an Komponenten ausgeschlossen werden, weil es mit einer "exotischen" Sprache nicht vertraut ist.

2 Spielregeln von *Fantastic Feasts*

Dieses Kapitel beschreibt die Spielregeln und den Ablauf einer Quidditch-Partie in *Fantastic Feasts*. Das eigentliche Spiel besteht aus einer rundenbasierten Partie des allseits beliebten Sports Quidditch. Der Rundenablauf ist bewusst relativ einfach gehalten (lock-step, vollständige Information, keine Nebenläufigkeit), um das Kommunikationsprotokoll und die KI-Logik nicht zu komplex werden zu lassen.

2.1 Spielfeld und Geometrie

Ein **Quidditch-Spielfeld** besteht aus einer ovalen Form, die in ein Raster von 17x13 quadratischen Feldern eingepasst ist (s. Abb. 2). Im Zentrum des Spielfeldes liegt der **Mittelkreis** (die 3x3 dunkelgrünen Felder), und darin das **Mittelfeld** *M*. An den gegenüberliegenden Enden des Spielfelds sind die jeweiligen **Hüterzonen** (rote Felder). In jeder Hüterzone sind **drei Torringe** *TR*.

Abb. 3 veranschaulicht Regeln für mögliche (Tor-)Schussvektoren und die Konzepte von überstrichenen Feldern und Entfernung.

Als **Schussvektor** bezeichnen wir den Pfeil vom Mittelpunkt eines Rasterfeldes zum Mittelpunkt eines anderen Rasterfeldes.

Alle Rasterfelder, die von einem Schussvektor geschnitten werden, bezeichnen wir als von diesem Vektor **überstrichene Felder.** "Geschnitten" bedeutet, dass Felder, die nur an einer Ecke tangiert werden (vgl. die beiden mit *X* markierten Felder) nicht als überstrichen gelten.

Die **Entfernung** zwischen zwei Feldern *A* und *B* ist definiert als die kleinste Zahl von Zügen von Feld zu Feld, die man machen muss, um von *A* zu *B* zu gelangen. Dabei darf man horizontal, vertikal und diagonal auf Nachbarfelder ziehen (so wie ein König im Schach ziehen kann). Man beachte dabei, dass im Allgemeinen *nicht* gilt, dass "Entfernung = überstrichene Felder - 1".

2.2 Bälle

Quidditch wird mit vier Bällen gespielt:

- Der **Quaffel,** ein roter Lederball, der von den Jägern durch die gegnerischen Torringe geworfen wird, um Punkte zu erzielen. Ein Tor gibt **10 Punkte.** Der Quaffel ist passiv, und bewegt sich nur, wenn er von einem Spieler mitgetragen, oder auf ein Zielfeld geworfen wird. Ein Jäger oder Hüter nimmt den Quaffel auf, wenn er auf das Feld zieht, auf dem der Quaffel liegt, oder wenn der Quaffel auf dem Feld zu landen kommt, auf dem der Spieler steht. Wenn ein Treiber oder Sucher auf das Feld des Quaffels zieht, oder der Quaffel auf dem Feld eines solchen Spielers landet, so hopst der Quaffel auf ein zufälliges freies Nachbarfeld.
- Zwei **Klatscher,** kleine schwarze Bälle, die eigenständig herumfliegen können, und versuchen Spieler vom Besen zu hauen. Ein Klatscher sucht sich jedesmal, wenn er mit seiner Rundenphase dran ist, den nächsten Spieler, der kein Treiber ist, aus und bewegt sich ein Feld auf ihn zu. Wenn der Klatscher es schafft, sich auf das Feld eines Spielers zu bewegen, wird dieser Spieler mit einer gewissen Wahrscheinlichkeit **ausgeknockt.**
- Der **Goldene Schnatz,** ein kleiner goldener geflügelter Ball, der eigenständig erratisch herumfliegt, und zu vermeiden versucht, von den Suchern gefangen zu werden. Historisch war das Fangen des Schnatzes 150 Punkte wert, aber da ein so hoher Wert das Spielergebnis fast immer dominiert (außer ein Team führt mit mehr als 15 Toren Abstand), wird in *Fantastic Feasts* die Punktewertung der International Quidditch Assocation [1] verwendet, nach der der Schnatz **30 Punkte** wert ist. In seiner Rundenphase achtet der Schnatz auf den nächsten Sucher, und wählt unter allen möglichen freien Nachbarfeldern, die eine größere Entfernung zu diesem Sucher haben als sein gegenwärtiges, ein zufälliges aus, und bewegt sich darauf. Falls es keine solchen Felder gibt, bewegt sich der Schnatz auf ein zufälliges freies Nachbarfeld (ggf. tunnelt er, wie in den allgemeinen Regeln beschrieben).

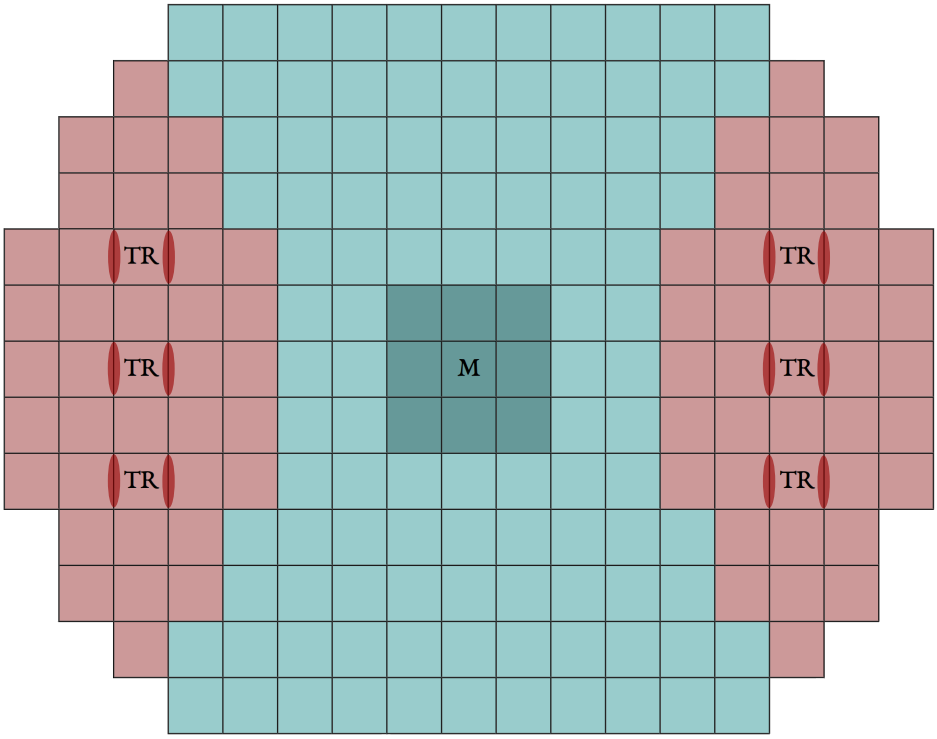

Abbildung 2: Das Spielfeld für Quidditch.

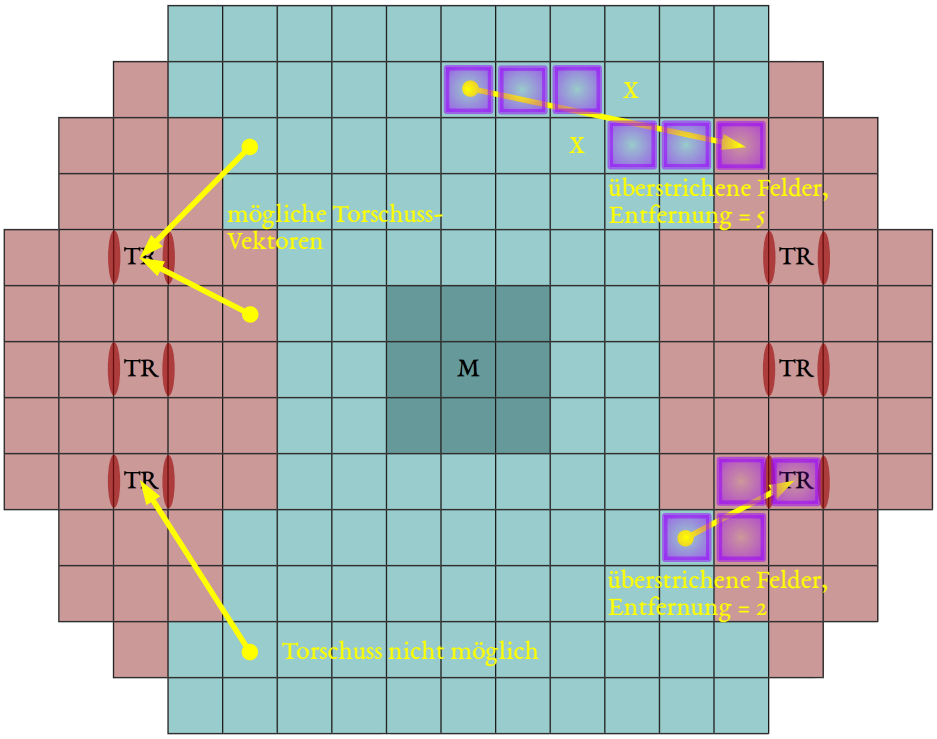

Abbildung 3: Überstrichene Felder, Entfernungen und Torschussvektoren.

2.3 Besen und Bewegung

Die **Besen** sind das wichtigste Sportgerät im Quidditch. In Partien von *Fantastic Feasts* werden die folgenden, vom Besenregulations-Kontrollamt zugelassenen, Sportflugbesen verwendet (in aufsteigender Qualität):

- Zunderfauch
- Sauberwisch 11
- Komet 2-60
- Nimbus 2001
- Feuerblitz

Bedingt durch den komplizierten Sponsoring-Rahmenvertrag gilt die **Besen-Repräsentanz-Regel:** jede der fünf Besenmarken muss von mindestens einem der sieben Spieler eines Teams verwendet werden. Mit anderen Worten, alle Besen müssen in der Ausrüstung eines Teams mindestens einmal vertreten sein.

In seiner Rundenphase kann sich ein Spieler immer ein Feld weit bewegen, und mit einer an seinen Besen geknüpften Wahrscheinlichkeit ein zweites Mal ein Feld weit ziehen. Je "besser" der Besen ist, desto höher ist diese Wahrscheinlichkeit, die in der Partie-Konfiguration definiert wird.

2.4 Teams

Bei einer Partie Quidditch treten **zwei Teams** gegeneinander an. Jedes Team sollte einen beeindruckend klingenden Namen haben, ein Team-Motto, ein Logo, und eine Hauptfarbe für die Trikots, sowie eine Ersatzfarbe, falls das Team gegen ein anderes mit ähnlichen Trikots antritt. Zur Inspiration findet man in [10] eine Abhandlung über historische Quidditchteams.

2.5 Spieler und Positionen

Ein Quidditchteam besteht aus **sieben Spielern,** die auf **vier Positionen** spielen:

- **Drei Jäger,** die versuchen den Quaffel durch die gegnerischen Torringe zu werfen, um Tore zu erzielen. Zu einem Zeitpunkt darf nur ein angreifender Jäger in der gegnerischen Hüterzone sein. Wenn ein Jäger auf ein Feld zieht, auf dem der Quaffel liegt, so nimmt er diesen damit auf. In seiner Rundenphase macht ein Jäger zuerst seine Bewegung, und dann ggf. einen Wurf mit dem Quaffel.
- **Zwei Treiber,** ausgestattet mit Kloppern, mit denen sie die beiden Klatscher von den eigenen Teamkameraden fernzuhalten und auf die Spieler des gegnerischen Teams zu lenken versuchen. Wenn ein Treiber auf das Feld eines Klatschers zieht oder bereits dort ist, so kann er in dieser Rundenphase danach den Klatscher kloppen. Er macht also einen "Wurf" mit dem Klatscher auf ein Zielfeld.
- **Ein Hüter,** der die eigenen Torringe schützt, und verhindern soll, dass gegnerische Jäger Tore werfen. Hüter dürfen den Quaffel aufnehmen und werfen, können aber selbst keine Tore erzielen.
- **Ein Sucher,** dessen einzige Aufgabe es ist, nach dem Goldenen Schnatz Ausschau zu halten, und ihn genau dann zu fangen, wenn sein Team mit den 30 Punkten, die der Schnatz bringt, gewinnt. Wenn immer der Sucher am Ende seiner Rundenphase auf dem Feld ist, auf dem der Schnatz liegt, fängt er ihn mit einer bestimmten Wahrscheinlichkeit.

Außerdem hat jeder Spieler einen Namen, ein Geschlecht, und einen Besen als Ausrüstung. Maximal vier Spieler eines Teams dürfen demselben Geschlecht angehören. Spieler machen in ihrer Rundenphase zuerst ihre Bewegung. Sie können stehenbleiben, ein Feld weit ziehen, oder versuchen nach dem ersten Feld noch ein Feld weiter zu ziehen. Anschließend können sie Aktionen durchführen wie den Quaffel zu werfen oder Klatscher zu kloppen.

2.6 Quaffel-Werfen und Klatscher-Kloppen

Jäger und Hüter können **Würfe mit dem Quaffel** machen. Um den Quaffel aufzunehmen, bewegen sie sich auf das Feld, auf dem er liegt, oder bekommen ihn ihrerseits zugeworfen. Ein Spieler, der den Quaffel werfen möchte, wählt zuerst ein Zielfeld. Nun werden alle vom Schussvektor überstrichenen Felder der Reihe nach betrachtet. Jeder gegnerische Spieler auf einem überstrichenen Feld hat eine bestimmte Wahrscheinlichkeit, den **Quaffel abzufangen.** Gelingt dies einem Hüter oder Jäger, so geht der Quaffel in dessen Besitz über. Im Falle eines Treibers oder Suchers, hopst der Quaffel auf ein zufälliges freies Nachbarfeld dieses Spielers.

Falls der Quaffel nicht abgefangen wurde, wird der **eigentliche Wurf abgehandelt,** der mit einer Wahrscheinlichkeit P^d erfolgreich ist, wobei P eine elementare Wurferfolgswahrscheinlichkeit ist, und d die Entfernung zum Zielfeld. Bei einem **erfolgreichen Wurf** landet der Quaffel auf dem Zielfeld. **Schlägt der Wurf fehl,** so wird der Quaffel auf einem zufälligen freien Feld in einem $n \times n$ Quadrat um das Zielfeld platziert, wobei $n = \lceil \frac{d}{7} \rceil$ (ein Siebtel der Entfernung, aufgerundet). Torringfelder gelten dabei als freie Felder (wenn kein Spieler daraufsteht), und Spieler können Tore oder Eigentore schießen, wenn der Quaffel im Rahmen eines missglückten Wurfs auf einem Torringfeld landet.

Für erfolgreiche **Torschüsse** gilt das Prinzip, dass der Spieler, der den Quaffel wirft, einen möglichen Torschussvektor haben muss (vgl. Abb. 3). Ein **Torschussvektor** ist nur dann gültig, wenn der Schussvektor durch eine der rot markierten Seiten in das Torringfeld läuft, wobei auch diagonal über die Ecke dieser Seite ein Tor erzielt werden darf (s. die beiden oberen Schussvektoren). Ein Schussvektor, der über eine der anderen Kanten in das Torringfeld eindringt, kann kein Tor erzielen (s. der untere Schussvektor).

Wenn der Quaffel auf einem Torringfeld zu liegen kommt (egal ob es als Tor zählt oder nicht), so geht er unmittelbar am Ende dieser Rundenphase in den Besitz des abwehrenden Hüters über, falls dieser sich in seiner eigenen Hüterzone befindet. Andernfalls wird der Quaffel auf das Mittelfeld gelegt, oder auf ein zufälliges freies Nachbarfeld, falls das Mittelfeld besetzt ist.

Ein Treiber, der auf dem Feld eines Klatschers ist, kann diesen **Klatscher kloppen.** Er wählt dazu ein Zielfeld derart, dass die Entfernung zu ihm höchstens 3 ist, und alle überstrichenen Felder zwischen dem Treiber und dem Zielfeld frei sind. Der Klatscher landet dann auf diesem Zielfeld. Steht dort ein Spieler, der von Klatschern angegriffen werden kann (Hüter, Jäger, Sucher) so wird dieser mit einer bestimmten Wahrscheinlichkeit **ausgeknockt.** Ein Klatscher, der einen Spieler ausgeknockt hat, zischt unmittelbar daraufhin davon, und wird auf einem zufälligen freien Feld platziert.

2.7 Fanblock

Quidditch lockt seit jeher Horden von ~~Hooligans~~ überenthusiastischen **Fans** an. Da *Fantastic Feasts* anstrebt, eine inklusive Sportveranstaltung zu sein, sind in den Stadien auch nicht-menschliche magische Wesen zugelassen. Leider hat das gelegentlich nachteilige Auswirkungen auf die sportliche Fairness, denn manche dieser Fans lassen sich in emotionalen Momenten dazu hinreißen, auf unerlaubte Weise (etwa mit Zaubersprüchen) ins Spielgeschehen einzugreifen.

Hier ist eine Liste magischer Kreaturen, und ihrer Zaubersprüche oder Fähigkeiten:

- **Elfen** können einen eigenen oder gegnerischen Spieler auf ein zufällig gewähltes freies Feld **teleportieren.**
- **Kobolde** können einen gegnerischen Spieler mit einem **Schockzauber** treffen. Der getroffene Spieler vertändelt zuerst den Quaffel, falls er ihn hat, und wird dann auf ein zufällig gewähltes freies Nachbarfeld gestoßen.
- **Trolle** können technisch gesehen nicht zaubern weil sie nicht sonderlich klug sind. Dafür können sie aber so laut **brüllen,** dass der Spieler, der den Quaffel hält, diesen vor Schreck vertändelt.
- **Niffler** sind ganz wild auf alle glänzenden Sachen. Ein Niffler kann **schnatzschnappen,** was dazu führt, dass der Schnatz eine Ausweichbewegung auf ein zufällig gewähltes freies Nachbarfeld macht.

Alle diese Eingriffe in das laufende Spiel werden, nachdem ihre Wirkung eingetreten ist, vom **Schiedsrichter** mit einer jeweils eigenen bestimmten Wahrscheinlichkeit erkannt, worauf der entsprechende Fan bestraft wird.

Jedes Team darf sieben Fans zu einer Partie einladen, wobei jede der vier Rassen mindestens einmal vertreten sein muss. Die Zusammenstellung des Fanblocks ist Teil der Quidditchteam-Konfiguration. Die Fans einer Mannschaft werden genauso vom Benutzer gesteuert, wie die Spielerfiguren.

2.8 Allgemeine Regeln, Aktionen, und Ereignisse

- Viele Aktionen und Ereignisse sind mit bestimmten **Erfolgs- oder Eintrittswahrscheinlichkeiten** verknüpft. Diese Wahrscheinlichkeiten werden in der Partie-Konfiguration definiert.
- Spieler und Bälle dürfen sich nur auf den Rasterfeldern des Spielfelds bewegen. Auf einem Feld kann höchstens ein Spieler gleichzeitig platziert sein. Bälle können gemeinsam auf einem Feld sein, inklusive der Felder, auf denen ein Spieler ist.
- Spieler machen in ihrer Rundenphase immer zuerst eine **Bewegung,** und danach eine **Aktion** wie z.B. den Quaffel zu werfen, usw.
- Ein Jäger darf als Aktion einem anderen Jäger auf einem Nachbarfeld, der den Quaffel hält, diesen entreißen. Das gelingt mit einer bestimmten Wahrscheinlichkeit. Ebenso darf ein Jäger einem Hüter den Quaffel entreißen, aber nur außerhalb der Hüterzone dieses Hüters.
- Ein Spieler, der von einem Klatscher ausgeknockt wird, vertändelt zuerst ggf. den Quaffel, und muss, wenn er das nächste Mal mit seiner Rundenphase drankommt (in dieser Runde oder der nächsten) aussetzen. Er hält also benommen seine Position, bewegt sich nicht, und macht auch sonst keine Aktionen wie z.B. Würfe oder den Schnatz fangen.
- Ein Jäger, der den Quaffel trägt kann diesen **vertändeln,** d.h. aus irgendeinem Grund verliert er die Kontrolle über den Ball, der dann auf ein zufälliges freies Nachbarfeld des Spielers platziert wird.
- Bei Regeln, die spezifizieren, dass etwa der Schnatz sich vom "nächsten" Sucher entfernen will, oder ein Klatscher sich auf den "nächsten" Nicht-Treiber zubewegt, wird bei gleichem Abstand zu den relevanten Spielern, einer zufällig als "nächster" gewählt. Analog werden ähnliche "Gleichstände", die sich nach irgendeiner Metrik ergeben, durch Zufallsentscheidung gebrochen.
- Wenn ein Spieler oder Ball auf "ein freies Nachbarfeld" gesetzt werden soll, aber alle acht umgebenden Felder von anderen Spielern besetzt sind, oder wegen des Spielfeldrandes nicht ausgewählt werden können, so wird rekursiv von einem zufälligen besetzten Nachbarfeld weitergesucht, bis irgendwann ein freies Feld gefunden wurde. Spieler und Bälle können also durch besetzte Felder "hindurchtunneln" um auf ein freies Feld gesetzt zu werden.

2.9 Fouls

Als fliegender Quasi-Vollkontaktsport bleibt es nicht aus, dass Quidditchspieler manchmal in einer Art und Weise aneinandergeraten, die nicht ganz regelkonform ist.

Das Folgende ist eine Liste der am häufigsten vorkommenden **Fouls,** mit relevanten Spielern (d.h. welche Position diese Art Foul begehen kann).

- **Flacken (alle Spieler):** Der abwehrende Spieler bewegt sich auf ein Torringfeld, und verhindert damit, das über diesen Torring ein Tor erzielt werden kann. Ein abwehrender Spieler darf direkt vor einem Torringfeld stehen, aber nicht darauf. Steht er darauf, ist die Wahrscheinlichkeit für den Schützen, ein Tor zu erzielen, null, und der Quaffel hopst sofort auf ein freies Nachbarfeld, falls er nach der Abhandlung des Wurfes auf dem Torringfeld gelandet wäre.
- **Nachtarocken (Jäger):** Ein Jäger erzielt mit 100% Erfolgswahrscheinlichkeit ein Tor, indem er, den Quaffel haltend, auf ein Torringfeld zieht, anstatt von einem Feld außerhalb des Torrings den Quaffel zu werfen, wie es die Regeln verlangen.
- **Stutschen (Jäger):** Ein angreifender Jäger fliegt in die gegnerische Hüterzone, während schon ein anderer angreifender Jäger dort ist. Ein Team darf zu jedem Zeitpunkt höchstens einen seiner Jäger in der gegnerischen Hüterzone haben.

- **Keilen (alle Spieler):** Absichtlich mit einem Gegner zusammenstoßen, indem man auf sein Feld zieht. Die Zielperson vertändelt ggf. zuerst den Quaffel, und wird danach auf ein zufällig gewähltes freies Nachbarfeld verdrängt.
- **Schnatzeln (alle Spieler, ausgenommen der Sucher):** Den Goldenen Schnatz berühren, wenn man kein Sucher ist. Dazu bewegt sich ein Spieler auf das Feld des Schnatzes, und verhindert damit ggf. dass ein Sucher dorthin ziehen kann, um den Schnatz zu fangen.

Alle diese Fouls werden, nachdem ihre Wirkung eingetreten ist, vom **Schiedsrichter** mit einer jeweils eigenen bestimmten Wahrscheinlichkeit erkannt, worauf der foulende Spieler entsprechend bestraft wird. Dabei wird also auf ein Foul folgend immer zuerst die unmittelbare Auswirkung umgesetzt, dann das eventuelle Erkennen durch den Schiedsrichter abgewickelt, und dann ggf. weitere Ereignisse, die aus der Aktion des mit dem Foul davongekommenen Spielers resultieren.

2.10 Schiedsrichter

Trotz allem Vertrauen auf die Sportlichkeit und Fairness von Spielern und Fans kann auf den Einsatz von **Schiedsrichtern** nicht verzichtet werden. Schiedsrichter sind keine Einheit, die auf dem Spielfeld dargestellt wird, sondern beobachten einfach die Partie und reagieren auf bestimmte Ereignisse.

Schiedsrichter geben sich zwar Mühe, sind aber leider nicht perfekt. Deshalb wird jedes **Foul,** das ein Spieler begeht, und jede **Einmischung** eines Fans mit einer spezifischen Wahrscheinlichkeit, die in der Partie-Konfiguration definiert ist, erkannt. Ein **erwischter Fan wird aus dem Stadion verwiesen** (verschwindet also für den Rest der Partie), ein **beim Foulen ertappter Spieler wird vom Platz verbannt, bis das nächste Tor fällt.** Sobald also in einer Runde ein Tor fällt, werden alle vom Platz verbannten Spieler am Ende dieser Runde (nach allen Rundenphasen) wieder aufs Feld gelassen. Sie dürfen sich dazu auf einem selbstgewählten freien Feld in der eigenen Hälfte platzieren. Sollte ein Client nicht innerhalb der dafür zugelassenen Zeit seine Wahl mitteilen, so platziert der Server die neu einzusetzenden Spieler dieses Clients zufällig in dessen Hälfte.

Sollten von einem Team am Ende einer Runde drei oder mehr Spieler vom Platz verbannt sein, so wird dieses Team unmittelbar **disqualifiziert,** und das andere Team gewinnt die Partie. Sollte diese Bedingung für beide Teams zutreffen, so endet die Partie mit dem Sieg desjenigen Teams, das mehr Punkte hat. Falls beide Teams gleich viele Punkte haben, gewinnt das Team, das *nicht* als erstes drei verbannte Spieler in dieser Runde hatte.

2.11 Start der Partie

Zu Beginn der ersten Runde dürfen sich die Spieler jedes Teams beliebig in der eigenen Hälfte des Spielfelds, jedoch außerhalb des Mittelkreises, platzieren. Der Quaffel und die beiden Klatscher beginnen auf dem Mittelfeld. Der Goldene Schnatz erscheint zu Beginn der dreizehnten Runde auf einem zufällig gewählten freien Feld, das möglichst gleich weit von beiden Suchern entfernt ist.

2.12 Runden und Rundenphasen

Das Spiel läuft in **Runden** ab, die in aufeinanderfolgende **Rundenphasen** unterteilt sind. Die Runden werden ab 1 gezählt. Die Rundenphasen einer Runde sind, in Reihenfolge:

1. **Ballphasen,** für die Bälle, die sich selbstständig bewegen können:
   (a) Der Goldene Schnatz macht seine Bewegung.
   (b) Die beiden Klatscher machen in zufälliger Reihenfolge ihre Bewegung.

2. **Spielerphasen:** Jeder Spieler eines Quidditchteams macht seinen Zug (optionale Bewegung, gefolgt von optionaler Aktion, z.B. Ballwurf, Foul), wobei sich die Teams abwechseln. In jeder Runde wird neu zufällig bestimmt, welches Team dabei beginnt. Die Reihenfolge der Spieler innerhalb eines Teams wird zufällig bestimmt.[2]
3. **Fanphasen:** Jeder Fan macht seinen Zug (optionale Einmischung in die Partie), wobei sich die Fanblöcke der Teams abwechseln. In jeder Runde wird neu zufällig bestimmt, welcher Fanblock dabei beginnt. Die Reihenfolge der Fans innerhalb eines Fanblocks wird zufällig bestimmt. Wenn in 2. oder 3. durch Platzverweise ein Team weniger Spieler oder Fans hat als ein anderes, so wird eben solange abgewechselt, bis nur noch ein Team Spieler/Fans übrig hat, die noch keinen Zug gemacht haben. Diese machen dann alle direkt hintereinander ihren Zug.

[2] Bsp.: Es wird ausgewürfelt, dass Team A beginnt. Ein zufällig gewählter Spieler von Team A macht seinen Zug, dann ein zufälliger von Team B, dann wieder ein zufälliger von A, usw. bis alle durch sind.

2.13 Siegbedingungen und Überlänge

Jede Partie Quidditch hat immer einen **eindeutigen Sieger.**

1. Bei **Disqualifikation** eines Teams gewinnt das andere Team.
2. Sobald ein Sucher in seiner Rundenphase den **Goldenen Schnatz fängt,** endet die Partie unmittelbar. Es gewinnt das Team, das dann am meisten Punkte hat.
3. Bei **Punktgleichstand** am Ende einer Partie gewinnt das Team, dessen Sucher den Goldenen Schnatz gefangen hat. (Die Überlängenbehandlung sorgt ja dafür, dass am Ende immer ein Sucher den Schnatz fängt.)

2.13.1 Behandlung überlanger Spiele

Um zu vermeiden, dass Partien übertrieben lange dauern (die längste aufgezeichnete Partie Quidditch dauerte drei Monate [10]), brauchen wir einen Mechanismus, der die **maximal mögliche Dauer einer Partie begrenzt.** Wenn die Partie über mehr Runden läuft, als ein in der Partie-Konfiguration festgelegter Höchstwert, wird sie durch ein verändertes Verhalten des Goldenen Schnatzes einem beschleunigten Ende zugeführt. Sobald diese **Überlängenbedingung** eintritt, wird die Wahrscheinlichkeit eines Suchers, den Schnatz auf einem Feld zu fangen, auf 100% gesetzt. Der Schnatz ist also müde und will nicht mehr ausweichen. Er ignoriert auch alle Effekte, die auf ihn ausgeübt werden sollen (bspw. wenn ein Niffler schnatzschnappen will). Sollte innerhalb der ersten drei Runden nach Eintritt der Überlänge der Schnatz noch nicht gefangen sein, so fängt der Schnatz an, zum Mittelpunkt des Spielfelds zu fliegen, wobei er Sucher ignoriert (also nicht mehr versucht von ihnen wegzukommen). Sobald der Schnatz das Mittelfeld erreicht hat, verharrt er dort. Sollte der Schnatz, nachdem er sich auf dem Mittelfeld zur Ruhe begeben hat, nach weiteren drei Runden immer noch nicht gefangen sein, fliegt er, sobald er mit seiner Rundenphase an der Reihe ist, aus Frustration innerhalb dieses Zuges direkt dem nächsten Sucher ins Gesicht, womit die Partie dann sofort endet, und der Schnatz als von diesem Sucher gefangen gilt.

3 Funktionalität von Komponenten

Die einzelnen Komponenten des Multiplayer-Spiels müssen bestimmte Funktionalitäten zur Verfügung stellen und Anforderungen erfüllen. Dabei wird zwischen **verbindlichen Anforderungen** und **optionalen Anforderungen** unterschieden. Optionale Anforderungen und selbst erdachte Features können nach Absprache mit dem Auftraggeber freiwillig implementiert werden, dürfen aber nicht zu Inkompatibilitäten mit Komponenten führen, die diese nicht beinhalten. Komponenten, Protokolle und Formate sollten ggf. so gestaltet werden, dass optionale Features ignoriert werden können, sofern Daten zu diesen in Nachrichten oder Dateien enthalten sind. Die Details regelt das Standardisierungskomitee.

3.1 Server

Verbindlich: Der Server lädt beim Start eine Partie-Konfiguration. Alle Einstellungen für eine Partie werden in der Partie-Konfiguration gespeichert. Enthaltene Konfigurationsdaten sind bspw. erlaubte Zeitspannen für Aktionen in den Rundenphasen, Rundenanzahl bis zum Eintritt der Überlängenbedingung für eine Partie, Wahrscheinlichkeiten von Ereignissen etc.

Verbindlich: Der Server erlaubt genau zwei Clients (Benutzer-Clients oder KI-Clients), sich über das Netzwerk bei ihm für eine Partie als Mitspieler anzumelden, und validiert ihre Quidditchteam-Konfigurationen.

Verbindlich: Sobald sich zwei mitspielende Clients beim Server registriert haben, startet der Server eine Partie und wickelt sie gemäß der Spielregeln ab.

Verbindlich: Der Server erlaubt Benutzer-Clients sich als Zuschauer für eine (ggf. bereits laufende) Partie zu registrieren. Sie bekommen dann den aktuellen Spielzustand und zukünftige Updates geschickt.

Verbindlich: Falls ein Client nicht innerhalb der in der Partie-Konfiguration festgelegten Zeitspanne dem Server seine Aktionen für eine Rundenphase mitteilt, nimmt der Server an, dass der Spieler keine Aktion in dieser Phase ausführen wollte (nach den Spielregeln ist dies immer möglich). Der Client wird übergangen, bleibt aber zunächst im Spiel. Verspätet eintreffende Nachrichten für eine Rundenphase, die in einer späteren Rundenphase beim Server eingehen, sollten entsprechend vom Server nicht als Protokollverletzung betrachtet, sondern als verspätet erkannt und einfach verworfen werden.

Verbindlich: Falls ein mitspielender Client eine Pausierung der Partie wünscht, unterbricht der Server diese, bis einer der Mitspieler anzeigt, dass er weiterspielen möchte. KI-Clients dürfen keine Pausen verlangen.

Verbindlich: Falls ein mitspielender Client seine Connection zum Server verliert, so bleibt seine Session zunächst bestehen. Der Client kann sich erneut mit dem Server verbinden, und seine Session fortsetzen. Der Client bekommt dazu vom Server den vollständigen aktuellen Spielzustand geschickt.

Verbindlich: Falls ein Client sich nicht an das Kommunikationsprotokoll hält, oder eine nach den Spielregeln unzulässige Aktion durchführen will, sendet der Server dem Client eine Nachricht über diesen Fehler, beendet die Verbindung zu ihm und schließt ihn damit vom weiteren Verlauf der Partie aus. Im Fall eines mitspielenden Clients gewinnt dadurch der gegnerische Mitspieler.

Verbindlich: Der Server informiert alle Clients (Mitspieler und Zuschauer) über die Aktionen der Spieler und die Ereignisse, die sich daraus ergeben haben, und den daraus resultierenden Spielzustand.

Verbindlich: Der Server überprüft am Ende der relevanten Rundenphase, ob ein Spieler gemäß der Siegbedingungen gewonnen hat. Wenn dies der Fall ist, beendet er die Partie und benachrichtigt alle Clients entsprechend.

Optional: Der Server schreibt alle Informationen in eine Logdatei, die zum Nachvollziehen des Partieverlaufs nötig sind. Dadurch können Clients ein Replay der Partie abspielen. Das Standardisierungskomitee legt das Format für Logs bzw. Replay-Dateien fest.

3.2 Benutzer-Client

Verbindlich: Der Benutzer-Client kann sich über das Netzwerk bei einem Server für eine dort angebotene Partie als Mitspieler registrieren. Er lädt und sendet dazu seine Quidditchteam-Konfiguration.

Verbindlich: Der Benutzer-Client hat einen Zuschauermodus, bei dem er sich als passiver Zuschauer bei einem Server für eine Partie registrieren, oder einer bereits laufenden beitreten kann.

Verbindlich: Der Benutzer-Client teilt dem Server mit, dass er ein von einem Menschen gesteuerter Client ist.

Verbindlich: Der Benutzer-Client visualisiert das Spielgeschehen mittels einer graphischen Oberfläche.

Verbindlich: Der mitspielende Benutzer kann über die graphische Oberfläche die in den jeweiligen Phasen einer Partie möglichen Aktionen vornehmen.

Optional: Der Benutzer-Client kann dem Benutzer Funktionen anbieten, die ihn beim Wählen und Durchführen von Aktionen unterstützen. Bspw. kann die Künstliche Intelligenz eingebunden werden, um Vorschläge für gute Aktionen zu liefern.

Optional: Der Benutzer-Client kann Funktionen anbieten, um dem Benutzer eine komfortablere Bedienung bei der Eingabe von Aktionen zu ermöglichen (z.B. Hotkeys).

Verbindlich: Die Abwicklung der einzelnen Rundenphasen werden vom Benutzer-Client animiert dargestellt. Dabei wird darauf geachtet, dass die Dauer der Animationen an die für Rundenphasen gewährte Zeit angepasst ist.

Verbindlich: Der Benutzer-Client ermöglicht dem Benutzer einen Wunsch auf Pausierung der Partie anzugeben. Wenn eine Partie pausiert ist (egal ob von einem selbst oder dem Gegner), kann ein Benutzer angeben, wenn er eine Wiederaufnahme des Spiels wünscht. Nur Benutzer-Clients dürfen eine Pausierung beantragen, KI-Clients nicht.

Verbindlich: Sollte die TCP-Connection, die der Websocket-Connection zugrunde liegt, abbrechen, so sollte sowohl ein Benutzer- als auch KI-Client versuchen, wieder eine Verbindung zum Server aufzubauen, und die beim Server noch anhängige Session fortzusetzen. Gerade im WLAN kann es vorkommen, dass ein System kurzzeitig offline ist. Das sollte nicht jedesmal zu einem Ausscheiden aus einer laufenden Partie führen (wichtig für Robustheit im Turnier). Die Lebensdauer einer Session kann sich also über mehrere Reconnections erstrecken.

Verbindlich: Am Ende einer Partie wird der Gewinner angezeigt.

Optional: Am Ende einer Partie werden Statistiken über den Partieverlauf angezeigt, bspw. für jedes Quidditchteam erzielte Tore, Ballkontakte, Flugstrecken, Fouls, Zaubersprüche der Fans, etc.

Optional: Der Benutzer-Client kann ein Replay einer gespielten Partie aus einer von einem Server erstellten Logdatei abspielen.

3.3 KI-Client

Verbindlich: Ein KI-Client muss nicht-interaktiv auf einem Linux-System über die Kommandozeile in Form eines Docker-Containers gestartet werden können. Dabei können in einer vom Standardisierungskomitee definierten Form die nötigen Argumente (IP und Port des Servers, Quidditchteam-Konfiguration, ...) übergeben werden, damit der KI-Client sich mit einem bestimmten Server verbinden und einer Partie beitreten kann. Nach dem Start findet keine Kommunikation zwischen der KI und deren Benutzer mehr statt. Die KI spielt völlig autonom.

Verbindlich: Die KI teilt dem Server mit, dass sie eine KI ist.

Verbindlich: KI-Clients dürfen selbst keinen Wunsch auf Pausierung einer Partie stellen, müssen aber beim Spiel gegen einen von Menschen gesteuerten Benutzer-Client mit Pausierungen umgehen können. KI-Clients erlauben ihrem menschlichen Gegner eine beliebig lange Pausierung, d.h. sie sprechen nie einen Wunsch nach Wiederaufnahme der Partie aus.

Verbindlich: Die KI bestimmt in allen Rundenphasen regelkonforme und möglichst sinnvolle Aktionen innerhalb der in der Partie-Konfiguration vorgegebenen Zeit, und teilt sie dem Server gemäß dem Nachrichtenprotokoll mit.

Optional: Über eine Konfigurationsdatei oder Kommandozeilenargumente können verschiedene Intelligenzstufen oder Strategien für die KI eingestellt werden.

Optional: Der KI-Client kann eine Schnittstelle anbieten, um in den Benutzer-Client eingebunden zu werden. Die KI-Logik kann dann benutzt werden, um dem Benutzer beim manuellen Spiel Aktionen vorzuschlagen.

3.4 Quidditchteam-Editor

Verbindlich: Das Standardisierungskomitee definiert ein JSON-Schema für Quidditchteam-Konfigurationen und Partie-Konfigurationen. Die mit dem Quidditchteam-Editor erstellten Konfigurationen werden in diesem Format in einer Datei gespeichert, und können zur weiteren Bearbeitung aus einer Datei geladen werden.

Verbindlich: Quidditchteam-Konfigurationen und Partie-Konfigurationen können über eine graphische Oberfläche erzeugt und bearbeitet werden.

Optional: Der Editor kann eine zufällig generierte gültige Quidditchteam-Konfiguration erzeugen.

Optional: Der Editor kann die vom Benutzer geladenen oder erstellten Konfigurationen validieren. Offensichtlich unzulässige oder unsinnige Werte werden werden für den Benutzer hervorgehoben.

4 Vorgaben zum Entwicklungsprozess

In diesem Kapitel wird beschrieben, wie das Entwicklungsprojekt ablaufen soll, und welche Regeln dabei eingehalten werden sollen. Zusätzlich sind natürlich auch die Aufgabenstellungen und Anforderungen in den Team-Meilensteinen zu beachten.

4.1 Begleitveranstaltung

In der Begleitveranstaltung zum Softwaregrundprojekt werden regelmäßig neue **Meilensteine** und **Aufgaben** vorgestellt und besprochen. Neben diesem Lastenheft stellen auch diese Aufgabenblätter Anforderungen des Auftraggebers an die Projektteams dar.

4.2 Moodle

Alle für das Softwaregrundprojekt wichtigen Informationen und Dokumente werden über **Moodle** bereitgestellt. Dazu gehören Aufgabenblätter für Einzel-Übungen und Team-Meilensteine, Referenzen, usw.

Die **Mailing-Funktion von Moodle** ist der hauptsächliche Kommunikationskanal in Richtung Projektteilnehmer, also sollten alle Teilnehmer sicherstellen, dass sie Moodle-Nachrichten zeitnah empfangen und lesen. Dazu kann ggf. eine Weiterleitung per E-Mail eingerichtet werden.

Im Moodle-Kurs gibt es auch ein Forum und Wiki, dass alle Teams benutzen können, um Themen von gemeinsamem Interesse zu diskutieren, Termine abzustimmen, etc. Projektteilnehmer können sich auch jederzeit mit Fragen oder Vorschlägen an uns (die "Auftraggeber") wenden.

4.3 Git und Gitlab

Jedes Team soll alle Artefakte, abgesehen von den Einzel-Übungsblättern, die lediglich über Moodle abgegeben werden, in seinem **Git Repository** (`pmgit.informatik.uni-ulm.de/sopra-18-19/teams/teamXX.git`) archivieren. Das gilt insbesondere für Source Code und jegliche Dokumentation. Die Abgabe von Team-Meilensteine erfolgt über Commits mit entsprechenden Tags.

Die Issue-Tracker Funktion von **Gitlab** dient in der Implementierungsphase als Scrum Board.

4.4 Qualitätssicherung und SonarQube

Wir verwenden das Tool SonarQube [8] zur Analyse der Qualität von Source Code. Jedes Teammitglied sollte den von ihm neu geschriebenen Code vor dem Push ins Repo damit analysieren. Teams sollten auch regelmäßig, etwa im Sprint Review, die Qualität der kompletten Codebase des Projekts begutachten, und die Behebung erkannter Probleme als Tasks für den nächsten Sprint einplanen. Erfahrungsgemäß amortisiert sich die Zeit, die *während* der Programmierphase in das Schreiben (und auch Aktualisieren) von Tests, in automatische Code-Analyse und manuelle Reviews gesteckt wird sehr schnell, weil man sonst am Ende sehr viel mehr Zeit investieren muss um eine Menge von spät erkannten Bugs in einem riesigen Haufen von Code mit schlechter Qualität zu beheben.

4.5 Docker

Um das vollständige verteilte Spiel zu verwenden, muss man meist Komponenten von verschiedenen Teams auf mehreren Plattformen zum Laufen bringen (ein Server, zwei spielende Clients (Benutzer-Clients oder KI-Clients), potentiell mehrere Beobachter-Clients). Diese Komponenten sind oft in unterschiedlichen Sprachen und unter verschiedenen

Betriebssystemen entwickelt worden. Damit sie zuverlässig und ohne aufwendige Build-Prozesse auf ggf. anderen Plattformen ausgeführt werden können, benötigen wir eine robuste und flexible Lösung zum Paketieren dieser Komponenten. Dazu verwenden wir das Containerization-Framework **Docker** [3]. Die Komponenten, die "headless" (also ohne graphische Oberfläche) betrieben werden (Server und KI-Clients) sollen mit allen Abhängigkeiten als Docker-Images paketiert werden. Dadurch können wir diese Komponenten bei teamübergreifenden Integrationstests[3] und Turnieren flexibel auf Linux- oder Windows-Rechnern starten.

4.6 Tutorien und Rolle der Tutoren

Jedes Team hat ein zugeordnetes **wöchentliches Tutorium** und einen **Tutor**. Der Tutor ist gegenüber dem Team ein **Vertreter des Auftraggebers.** In den Tutorien werden die vom Tutor korrigierten Abgaben von Einzel-Übungsblättern und Team-Meilensteinen besprochen. Die Tutorien sind auch dazu da, Verständnisprobleme zu beheben, als Einzelperson oder Team Feedback zu allen Aspekten des Projekts zu bekommen, oder jegliche Herausforderungen und Schwierigkeiten zu besprechen. Für die Tutorien besteht Anwesenheitspflicht.

In der Implementierungsphase, während der ein an Scrum orientierter Entwicklungsprozess durchgeführt werden soll, übernehmen die Tutoren gegenüber dem Team auch die Rolle des **Product Owners.** Die Tutorien erfüllen dann auch die Rolle von **Sprint Meetings.** Auch dafür besteht Anwesenheitspflicht.

4.7 Agiler Entwicklungsprozess

Im Softwaregrundprojekt wird während der Implementierungsphase ein an **Scrum** orientierter agiler Prozess durchgeführt. Der Tutor übernimmt die Rolle des **Product Owners.** Es finden **Sprints** mit Dauer von zwei Wochen statt. Entsprechend gibt es **Sprint Planning, Sprint Review und Sprint Retrospective Meetings.**

Typischerweise findet an einem Tutoriumstermin zuerst Review und Retrospective für den vergangenen Sprint statt, und dann Planning für den kommenden. Dennoch sollte man diese Arten von Meetings mit ihrem jeweils eigenen Inhalt und Ziel nicht vermischen, sondern gedanklich klar trennen, und sauber der Reihe nach durchführen. Wir legen dabei Wert auf eine ordentliche Einhaltung des Scrum-Prozesses. Dazu gehört auch ein sinnvoller Umgang mit zentralen Artefakten wie z.B. dem Scrum-Board. Das Projekt soll am Ende nicht nur ein funktionierendes Produkt ergeben, sondern dieses soll auch in einem geregelten Entwicklungsprozess entstehen, um eine hohe Qualität und ein funktionierendes Projektmanagement zu gewährleisten.

4.8 Dokumentation

Jedes Team führt ein gemeinsames **Projekttagebuch.** Darin trägt jedes Team-Mitglied für sich selbst (!) alle Tätigkeiten und den dazugehörenden Zeitaufwand ein, den es für das Projekt erbracht hat. Dazu gehören Zeiten fürs Besuchen von Tutorien, den später stattfindenden Sprint Meetings, sonstige Besprechungen im Team, das Bearbeiten von Team-Meilensteinen, Aufwand für selbstständiges Einarbeiten in Technologien und Frameworks, Implementierungstätigkeiten, Zeiten für die Erstellung von Dokumenten – einfach alles was Sie für das Team-Projekt im Softwaregrundprojekt tun. Die Zeiten für das Besuchen der Softwaretechnik-Vorlesung und der Sopra-Begleitveranstaltung sollen jedoch nicht im Projekttagebuch eingetragen werden. Eine Vorlage für das Projekttagebuch wird im Moodle bereitgestellt.

Verfassen Sie ein **Entwicklerhandbuch** zum Produkt. Es soll die Architektur und die implementierte Funktionalität beschreiben. Listen Sie auch auf, welche Technologien und Frameworks Sie benutzen, und die Gründe dafür. Halten Sie außerdem alle wichtigen Entscheidungen, die im Team oder gemeinsam mit dem Tutor getroffen werden, fest.

Selbstverständlich muss auch der geschriebene Source Code ausführlich **kommentiert** werden. Maßnahmen zur **Qualitätssicherung** wie Code-Analyse, Reviews oder Tests müssen auch nachvollziehbar dokumentiert werden.

Für die Komponenten des Spiels soll jeweils ein **Benutzerhandbuch** erstellt werden, das ihre Funktionalität und

[3] a.k.a. LAN-Parties

Verwendung erklärt.

All dies ist besonders für die auf der Messe verkaufte Komponente wichtig, um den Käufer-Teams eine leichte Einarbeitung und Weiterentwicklung zu ermöglichen. Alle Dokumente sollen in sinnvoller Verzeichnisstruktur im Git Repository des Teams abgelegt werden.

4.9 Standardisierungskomitee

Fantastic Feasts ist ein verteiltes System. Das stellt Anforderungen an die Interoperabilität der verschiedenen Komponenten, wie Benutzer-Clients, KI-Clients, Servers und Quidditchteam-Editoren. Jeder Client muss mit jedem Server über das Netzwerk kommunizieren können. Alle Clients und Server müssen die von jedem Quidditchteam-Editor erstellten Quidditchteam-Konfigurationen verarbeiten können.

Das bedeutet, um *Fantastic Feasts* betreiben zu können, müssen ein gemeinsames **Nachrichtenprotokoll** und gemeinsame **Formate für Quidditchteam- und Partie-Konfigurationen** definiert werden, die von allen Teams in ihren Komponenten implementiert werden. Um das zu erreichen, wird ein teamübergreifendes **Standardisierungskomitee** gebildet. Jedes Team bestimmt unter seinen Mitgliedern einen **Standardisierungsbeauftragten,** der in das Standardisierungskomitee entsandt wird. Für die Sitzungen des Komitees besteht Anwesenheitspflicht. Falls der Beauftragte eines Teams verhindert ist, so liegt es in seiner Verantwortung einen Vertreter zu schicken.

Beim konstituierenden Treffen des Komitees wählt dieses einen **Vorsitzenden.** Der Vorsitzende hat die Aufgabe, Termine für die Sitzungen des Komitees zu planen, und diese zu leiten. Bei jeder Sitzung wird ein **Protokollführer** bestimmt, der den Ablauf der Sitzung und die dort getroffenen Entscheidungen schriftlich festhält. Die Sitzungsprotokolle und alle Standardisierungsdokumente werden den Teams über ein Git Repository des Komitees zur Verfügung gestellt.

Um bestimmte, abgegrenzte Themen zu diskutieren oder Entwürfe für Protokolle und Formate zu erstellen, kann das Standardisierungskomitee kleinere **Arbeitsausschüsse** einrichten. Die von den Ausschüssen erarbeiteten Entwürfe werden auf Sitzungen des ganzen Komitees zur **Abstimmung** gestellt, und werden durch Mehrheitsbeschluss angenommen. Der Auftraggeber hat dabei eine beratende Rolle und ein Vetorecht.

Die vom Standardisierungskomitee definierten Kommunikationsprotokolle und Dateiformate werden in **Standardisierungsdokumenten** beschrieben, die nach Verabschiedung im Komitee den Teams in der Implementierungsphase zur Verfügung gestellt werden. Falls es durch geänderte Anforderungen oder durch Erkenntnisse bei der Implementierung nötig wird, die Standards anzupassen, zu erweitern, oder Fehler und Widersprüchlichkeiten zu beheben, soll das Komitee diese Dokumente zeitnah überarbeiten.

4.10 Messe

Jedes Team implementiert einen **Benutzer-Client** und einen **KI-Client.** Von den übrigen beiden Komponenten, **Server** und **Quidditchteam-Editor,** wählt jedes Team *eine* aus, die es implementiert. Die andere, fehlende Komponente, die noch benötigt wird, um das komplette Multiplayerspiel zu haben, "kaufen" die Teams dann auf einer Messe von einem anderen Team ein.

Die **Messe** wird ungefähr zur Halbzeit des Projekts stattfinden. Jedes Team muss bis dahin seinen Server bzw. seinen Quidditchteam-Editor als fertiges, lauffähiges Package haben. Dazu gehören auch eine umfassende Dokumentation (Benutzer-/Entwicklerhandbücher, Entwurfsdiagramme, etc.) sowie Unittests. Auf der Messe präsentiert dann jedes Team an seinem Stand seine Komponente den anderen Teams, denen diese Komponente noch fehlt. Für jede "verkaufte" Lizenz bekommt das Team entsprechend Punkte für den Highscore gutgeschrieben.

Umgekehrt muss sich jedes Team die ihm noch fehlende Komponente unter den von anderen Teams angebotenen aussuchen. Nach dem "Verkauf" geht die Komponente mit Source Code und aller Dokumentation, Tests, usw. in den Besitz des Käufers über, der ab da ganz allein für die Wartung und ggf. Weiterentwicklung verantwortlich ist. Das bedeutet, der Käufer macht einen "hard fork". Da die Teams also mit gekauften Komponenten selbst weiterarbeiten müssen, sollte beim "Kauf" entsprechend Augenmerk darauf gelegt werden, eine gut designte, wart- und erweiterbare

Komponente mit hoher Testabdeckung und guter Dokumentation zu erwerben. Die Programmiersprache sollte natürlich auch eine sein, mit der das erwerbende Team vertraut ist.

4.11 Abschlussturnier

Am Ende des Projektzeitraums wird ein **Abschlussturnier** stattfinden, in dem die **KI-Clients** der Teams gegeneinander antreten. Das Turnier läuft in zwei Phasen ab. Zuerst findet eine Vorauswahl statt, um die besten KI-Clients zu bestimmen. Danach treten diese in einem öffentlichen Abschlussturnier in Finalrunden im K.O.-System gegeneinander an. Die Partien werden vom Publikum über Benutzer-Clients im Zuschauer-Modus verfolgt. Dafür werden die schönsten Benutzer-Clients von den Tutoren ausgewählt. Das Turnier ist eine Gelegenheit für die Teams, ihr Produkt vor einem größeren Publikum zu präsentieren, und in einem spannenden Wettbewerb ihre geistigen Kräfte und Coding Skills aneinander zu messen. Für Studierende aus niedrigeren Semestern, die das Softwaregrundprojekt später selbst noch absolvieren müssen, ist das Turnier ebenfalls interessant, um einen Eindruck vom Ergebnis des Projekts bekommen.

4.12 Highscore und Preise

Während des Softwaregrundprojekts wird ein **Highscore** geführt. Jedes Team bekommt Highscore-Punkte für die Leistung seiner Mitglieder bei Einzel-Übungsblättern und Team-Meilensteinen, entsprechend der Bewertungen durch die Tutoren. Es gibt auch Punkte für die auf der Messe "verkauften" Lizenzen.

Darüberhinaus behalten wir uns vor, bei herausragenden Leistungen oder besonderem Engagement im Projekt Achievements zu vergeben, die mit zusätzlichen Punkten in den Highscore eingehen.

Am Ende des Softwaregrundprojekts werden zwei Preise vergeben:

- Der **Highscore-Preis** für die beste Leistung über die gesamte Projektdauer hinweg bei Abgaben und sonstigen Aufgaben.
- Der **Turnier-Preis** für den Sieg im Abschlussturnier.

Den Gewinnerteams der beiden Preise winkt ewiger Ruhm und vielleicht sogar Reichtum...

5 Kriterien für die Abnahme

Um als Team eine erfolgreiche Projektabnahme zu erreichen und für das individuelle Bestehen des Softwaregrundprojekts müssen folgende Bedingungen erfüllt sein:

- Fristgerechte und korrekte Abgabe aller Team-Meilensteine mit ausreichender Bewertung.
- Fristgerechte Abgabe und Bestehen von mindestens n-1 der n Einzelübungsblätter.
- Regelmäßige Anwesenheit und Mitarbeit im Tutorium.
- Aktive Mitarbeit im Team. Dies wird durch das Projekttagebuch dokumentiert.
- Eigene Programmiertätigkeit in ausreichendem Umfang.
- Fristgerechte erfolgreiche Abnahmesitzung zum Ende des Projekts. Dazu gehört:
  - Vorlage einer lauffähigen Version der Software.
  - Umsetzung der Kernfunktionalität, d.h. aller verbindlichen Anforderungen.
  - Gute Qualität der entwickelten Software, mit sinnvoller Testabdeckung und nachgewiesen durch umfangreiche und dokumentierte Maßnahmen zur Qualitätssicherung.
  - Einhaltung des Scrum-Entwicklungsprozesses und sinnvolle Verwendung von Team-Repository und Scrum-Board im Gitlab.
  - Ordentliche Dokumentation der Software und des Projektverlaufs. Das umfasst insbesondere die zum Scrum-Prozess gehörenden Dokumente (Scrum-Board etc.).
- Anwesenheit beim Abschlussturnier.

5.1 Abnahmesitzung

Am Ende des Sommersemesters 2019 findet für jedes Team eine Abnahmesitzung statt, bei dem die entwickelte Anwendung dem Auftraggeber präsentiert und ihre Features demonstriert werden. Dabei soll jedes Teammitglied seinen persönlichen Anteil am Ergebnis präsentieren.

Beim Abnahmetreffen werden auch die geforderten Dokumente abgegeben.

And be careful with Baruffio's Brain Elixir...

Literatur

[1] International Quidditch Association. *IQA Rulebook 2018-2020*. 2018.

[2] Raphael Crawford-Marks, Lee Spector, and Jon Klein. Virtual witches and warlocks: A quidditch simulator and quidditch-playing teams coevolved via genetic programming. In *Late-Breaking Papers of GECCO-2004, the Genetic and Evolutionary Computation Conference. Published by the International Society for Genetic and Evolutionary Computation*, 2004.

[3] Docker, Inc. Docker. `https://www.docker.com/`. [Online; accessed 2018-11-20].

[4] IETF. RFC 6455: The WebSocket Protocol. `https://tools.ietf.org/html/rfc6455`. [Online; accessed 2018-11-20].

[5] IETF. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. `https://tools.ietf.org/html/rfc8259`. [Online; accessed 2018-11-20].

[6] Julie McDonough. *Muggles, and quidditch, and squibs, oh my! A study of names and onomastic wordplay in translation, with a focus on the Harry Potter series*. PhD thesis, University of Ottawa (Canada), 2004.

[7] Rachel Pennington, Ashley Cooper, Evan Edmond, Alastair Faulkner, Michael J Reidy, and Peter SE Davies. Injuries in quidditch: a descriptive epidemiological study. *International journal of sports physical therapy*, 12(5):833, 2017.

[8] SonarSource S.A. SonarQube. `https://www.sonarqube.org/`. [Online; accessed 2018-11-20].

[9] Newt Scamander. *Fantastic Beasts and where to find them*. Obscurus Books, 2001.

[10] Kennilworthy Whisp. *Quidditch through the Ages*. Whizz Hard Books, 2001.