THM | CAMPUS GIESSEN | MNI
TECHNISCHE HOCHSCHULE MITTELHESSEN
Mathematik, Naturwissenschaften und Informatik

Bachelorthesis

Konzeption und Umsetzung von Matchmaking Architekturen in Online-Spielen

zur Erlangung des akademischen Grades

Bachelor of Science

eingereicht im Fachbereich Mathematik, Naturwissenschaften und Informatik an der Technischen Hochschule Mittelhessen

von

Leon Schäfer

18. Januar 2022

Referent: Dr. Dennis Priefer Korreferent: < noch nicht festgelegt >

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Gießen, den 18. Januar 2022 Leon Schäfer

Abstract:

Der Einstieg in die Multiplayer-Spieleeprogrammierung gestaltet sich für angehende Entwickler als schwierig. Ein erstes eigenes Projekt umzusetzen erfordert viel Einarbeitung und es gibt viele Einstiegshürden. Insbesondere der Teil des Matchmakings ist ohne Budget oft schwierig umzusetzen. Es gibt schlichtweg zu wenig Einstiegshilfen in diesem Teil, da jedes Spiel sehr individuelle Anforderungen mit sich bringt.

Das Ziel dieser Arbeit ist, eine Einstiegshilfe für das Matchmaking in Multiplayer-Projekten zu geben. Hierzu wird eine abstrakte Vorlage konstruiert, welche auf viele Multiplayer Use-Cases anwendbar ist.

Dazu wurden abstrahierte Konzepte erstellt, welche das Matchmaking in Multiplayer Spielen abbilden und auf möglichst viele Spiele anwendbar sein sollen. Diese Konzepte bestehenden aus einzelnen Software-Komponenten, welche selbst noch keine konkrete Implementierungsvorgabe sind, sondern lediglich das Konzept der Komponente abstrakt beschreiben.

Als Proof of Concept wurden die beschriebenen Konzepte anhand einer Implementierung in einem Multiplayer-Spiel in der Unity Engine, mithilfe des Mirror Frameworks implementiert.

Ideensammlung für Gliederung::

1. Einführung:

Matchmaking –> Moderne Matchmaking-Architekturen/Algorithmen von heute (Beispiele) –> Elo-Zahl basiertes Matchmaking (Rangliste) + Lobby –> Generell: Parameterbasiertes Matchmaking –> Server Browser –> Lobbys –> ... –> Implementierungsbeispiele der "realen Welt"

2. Ableitbare Konzepte:

–> Offline/Online Verwaltung durch Network-Manager –> verwaltet u.a. Abbruchbedingungen (Maneull / Automatisch) für Client und Server –> Lobby-Manager(Verwaltung der Spieler in Lobby-Szenen) (optional) –> Prepare-InGame-Manager (Initialisierung v. Spawn Punkten, Objektspawn v. abh. Objekten) –> In-Game-Player-Manager (Verwaltung der Spieler in Spiel-Szenen) –> In-Game-Object-Manager (Verwaltung v. Objekt und Spieler-Respawn) –> In-Game-Progress-Manager (Verwaltung v. Spielfortschritt, u.a. zuständig für Spielbeendung) –> Dieser ist ebenfalls dafür zuständig, nach Beendigung einer Spiel-Session die Spieler in die öffline-Szeneöder zurück in ihre "Lobby-Szeneßu

befördern. Sollte die Implementierung eine Online-Lobby unterstützen, übernimmt im Anschluss wieder der "Pre-Game-Player-Manager"

–> HTTP Server / REST API / Dedicated Server –> Unterscheidung Hosting (Client Host / Dedicated Server Host) –> Einordnung der generalisierten Konzepte für das Matchmaking

3. Implementierung 1 :

Genutzte Frameworks / Technologien

Grundlagen des Mirror Frameworks: –> worauf basiert es eigendlich? –> Auszug Features | diese erklären. –> Network-Manager –> NetworkManager Callbacks –> Network Idendity / Network Behavior / Network Transform –> Network Behavior Callbacks –> Server und Clientcode in einer Datei –> Dedicated Server vs. Self-Host –> Synchronization –> Remote Actions –> Player Game Objects –> Anticheat –> Transports

4. Implementierung 2:

–> Beispiel Implementierung Matchmaking System (Hide n seek) 5. Abschluss:

Inhaltsverzeichnis

1 Einführung

Die Spieleindustrie ist im Wandel. Reine Singleplayer-Spiele, die ohne andere menschliche Mitspieler auskommen, werden immer seltener entwickelt. Die Erfolgsaussichten erscheinen um ein Vielfaches schlechter. Die drei bestplatzierten Titel der meistverkauften PC-und Konsolenspiele 2019 (FIFA 2020, Call of Duty: Modernware und Mario Kart 8 Deluxe) in Deutschland enthalten einen großen Multiplayer-Anteil.[gam20]

Deutlicher wird der Trend, wenn die Statistik der meistgespielten Titel auf der Online-Spiele-Vertreibsplattform Steam betrachtet wird. Das mit Abstand meistgespielte Spiel 2020 ist das Multiplayer-First-Person-Shooter Spiel 'Counter-Strike: Global Offensive' gefolgt von 'DOTA 2' auf Platz 2. Beide Spiele sind reine Multiplayer-Titel.[Git21]

Einer internationalen Umfrage zufolge geben 60% der Befragten an, seit Beginn der Corona Pandemie mehr Multiplayer-Videospiele zu spielen. [Sim20]

Wegen der Marktentwicklung ist es für angehende Entwickler, welche in der Spieleindustrie Fuß fassen möchten, auf lange Sicht eine gute Entscheidung, sich mit dem Bereich Multiplayer-Spieleentwicklung auseinanderzusetzen.

1.1 Problembeschreibung und Motivation

Konzeption und Entwicklung von Online-Multiplayer-Spielen stellt angehende Entwickler vor große Herausforderungen. Einer der Hauptunterschiede zwischen einem Multiplayer- und einem Einzelspieler-Spiel ist, dass eine Spielumgebung unabhängig von einem Spieler existiert. Beispielsweise benötigt das Versenden von Nachrichten über das Internet eine gewisse Zeit. Der Entwickler muss also herausfinden, wie man mit asynchroner Kommunikation, Clients und Servern umgeht. Ebenso stoßen Entwickler vor Probleme wie Hosting, Matchmaking und Datenverwaltung. [Pay19]

Die Einstiegshürden sind groß, der Dschungel an Technologien und Frameworks unübersichtlich. [MFa21]

Einheitliche Vorgehensmodelle und Best-Practises gibt es nicht oder beziehen sich stets auf eine spezielle Art von Spieltyp. Das Thema Hosting ist ebenfalls ein komplexer

Aspekt eines solchen Projekts, welcher von Anfängern oft nicht sofort durchschaut wird. Durch erste Projekterfahrung gewinnen die Entwickler nach und nach das Know-How, ohne ein erfahrenes Team sind die Chancen auf Misserfolg jedoch hoch. [Pay19]

Die Motivation für diese Arbeit war es diese Probleme zu untersuchen und Lösungsmöglichkeiten anhand von praktischen Konzepten zu schaffen. Angehende Entwickler können diese Konzepte in ihrem Praxisprojekt ausprobieren und idealerweise davon profitieren.

1.2 Ziele dieser Arbeit

Die Arbeit setzt sich folgende Ziele:

Um einen einfachen Einsteig in die Multiplayer-Spieleentwicklung zu gewährleisten, beschäftigt sich diese Arbeit mit einigen Grundlagen und bietet Ansätze, die Entwicklern einen Vorteil bei der Gestaltung ihrer Software-Architektur bringen soll.

Ein Entwickler soll außerdem nach dem Studium dieser Arbeit eine konkrete Vorstellung davon besitzen, welche Voraussetzungen ein Framework oder eine Game-Engine erfüllen muss, damit die Konzepte möglichst leicht umzusetzen sind.

Die Arbeit soll eine Blaupause liefern, welche andere Entwickler nutzen können, um einen leichteren Einstieg in ein Multiplayer-Projekt zu gewährleisten. Diese Blaupause soll allerdings derart generisch sein, dass sie unabhängig von einem konkreten Implementierungskontext funktioniert. Die beschriebenen Konzepte sollen auf möglichst viele Szenarien anwendbar sein. Der Fokus hier soll auf dem Matchmaking liegen. Auf den Begriff Matchmaking wird zu einem späteren Zeitpunkt genauer eingegangen.

Durch die Konzepte sollen verschiedene Use cases in der Multiplayer-Spieleprogrammierung abgedeckt werden. Beispiele sind das Aufbauen und Trennen von Client/Server Verbindungen, Matchmaking und Spielerverwaltung.

Als 'Proof-of-Concept' soll ein Prototyp eines Hide-and-Seek Multiplayer-Spiels dienen, welches sich an den abstrahierten Konzepten orientert, bzw. diese implementiert.

Folgende Forschungsfragen soll die Arbeit beantworten:

F1: Können durch die Untersuchung der beschriebenen Probleme Konzepte abgeleitet werden, die das Potential haben, diese Probleme zu lösen beziehungsweise zu vereinfachen?

F2: Welche Einstiegshürden gibt es? Wobei würden sich angehende Entwickler mehr Unterstützung wünschen?

1.3 Vorgehensweise

F1 soll wie folgt beantwortet werden:

Zunächst wird die Problemstellung anhand von Literatur und Internetquellen aufgezeigt und erläutert.

Aus der bisher gesammelten Praxiserfahrung und gefundenen Literatur werden Konzepte abstrahiert, welche jeweils einen bestimmten Einsatzzweck erfüllen sollen. Hierbei wurde auf Implementierungsdetails verzichtet und lediglich ein generelles Konzept erarbeitet, an welchem der Entwickler sich während des Implementierungsprozesses orientieren kann.

Jedes dieser Konzepte wurde in einem Prototyp umgesetzt, welche im Kapitel Realisierung näher vorgestellt wird.

F2 soll wie folgt beantwortet werden:

Parallel wird begonnen, eine Umfrage zu erstellen. Dieser beinhaltet konkrete Fragen an Multiplayer-Spieleentwickler. Die Fragen beziehen sich besonders auf mögliche Einstiegsschwierigkeiten bei der Multiplayer-Spieleentwicklung. Es werden angehende Entwickler in der Industrie sowie an der Hochschule befragt. Diese Umfrage wird gegen Ende der Arbeit ausgewertet.

1.4 Abgrenzung

Die Arbeit ist jedoch keine Schritt-für-Schritt Anleitung für Multiplayer-Spieleprojekte. Implementierungsdetails müssen von einem Entwickler, welcher diese Arbeit als Hilfsmittel nutzt, selbstständig konzipiert und umgesetzt werden. Es wird außerdem lediglich auf Spielkonzepte Rücksicht genommen(insbesondere Matchmaking), welche in der heutigen Industrie gängig sind[Wik21c]. Es ist durchaus denkbar, dass zukünftige Multiplayer/Matchmaking Architekturen nicht mehr mit den hier beschriebenen Konzepten umzusetzen sind.

Es ist außerdem nicht final bewiesen, dass die beschriebenen Methoden tatsächlich einen Mehrwert bei unterschiedlichen Projekten bieten. Die Konzepte wurden lediglich an einem Prototyp getestet und validiert. Um zu testen, ob die Konzepte auch praxistauglich

für verschiedene Arten von Projekten sind, muss eine weitergehende Untersuchung stattfinden.

Die erarbeiteten Konzepte sind außerdem unabhängig von einer Game-Engine oder einem Framework. Durchaus ist es aber möglich, dass ein bestehendes Framework oder eine Game-Engine Hilfestellung bei der Implementierung der Konzepte bietet.

1.5 Struktur der Arbeit

Diese Arbeit besteht aus der Einführung, in welcher die generelle Problemstellung erläutert, und die Forschungsfragen gestellt wurden. Ebenfalls wurde die Relevanz des Themengebiets verdeutlicht und eingeordnet.

Im Anschluss wird der Hintergrund erklärt, dort sind alle Informationen zu finden, um die späteren Konzepte zu verstehen. Hier werden ebenfalls Begrifflichkeiten und Grundlagen erläutert.

Das Kapitel Konzepte behandelt nun die erarbeiteten Konzepte, welche in Summe die oben beschriebene Blaupause abbilden. Es wird auf jedes einzelne Konzept im Detail eingegangen und genauer erklärt.

Anschließend werden die beschriebenen Konzepte in einem Prototyp umgesetzt, dieser wird im Kapitel Realisierung näher erklärt. In diesem Kapitel befindet sich kommentierter Beispielcode aus dem umgesetzten Prototyp um einen besseren Eindruck der gelösten Probleme zu bekommen.

In der Zusammenfassung werden die Ergebnisse der Arbeit beleuchtet und ein Ausblick aufgezeigt.

2 Hintergrund

2.1 Grundlagen des Networkings in Online-Videospielen

Die Grundlagen des Networkings in Videospielen lassen sich in zwei große Bereiche kategorisieren. Die physische und die logische Plattform.

Die physische Plattform setzt sich aus den physikalischen Komponenten zusammen, die zusammen eine Infrastruktur bilden. Hierzu zählt Hardware, welche in Rechenzentren eingesetzt wird, das lokale Endgerät wie z.B. ein Smartphone oder ein Personal Computer. Ebenfalls zählen Kabelleitungen und drahtlose Übertragungswege dazu.

Online-Videospiele sind aus technischer Sicht auch nichts anderes als Anwendungen die miteinander kommunizieren. Die physischen Restriktionen wie Bandweite und Latenz können also ebenfalls auf den Kontext der Multiplayer-Spiele angewendet werden. Die Menge der Informationen, welche über ein Netzwerk versendet werden kann je nach Spieltyp sehr hoch skalieren, weshalb sich die Entwickler eines Multiplayer-Spiels intensiv mit der logischen Plattform beschäftigen müssen.

Die logische Plattform baut auf der physischen Plattform auf und nutzt ihre Ressourcen. Die logische Plattform eines Online-Multiplayer Spiels kann unterteilt werden in Kommunikation zwischen Clients und Server, Datenspeicherung und Transfer, und Kontrolle des Spielflusses. Auf diese drei Punkte wird in den folgenden Sektionen näher eingegangen.

[Sme02b]

2.2 Verschiedene Arten des Hostings

Die Kommunikation zwischen Clients und Server kann in zwei verschiedenen Varianten vorkommen:

"Client-Host":

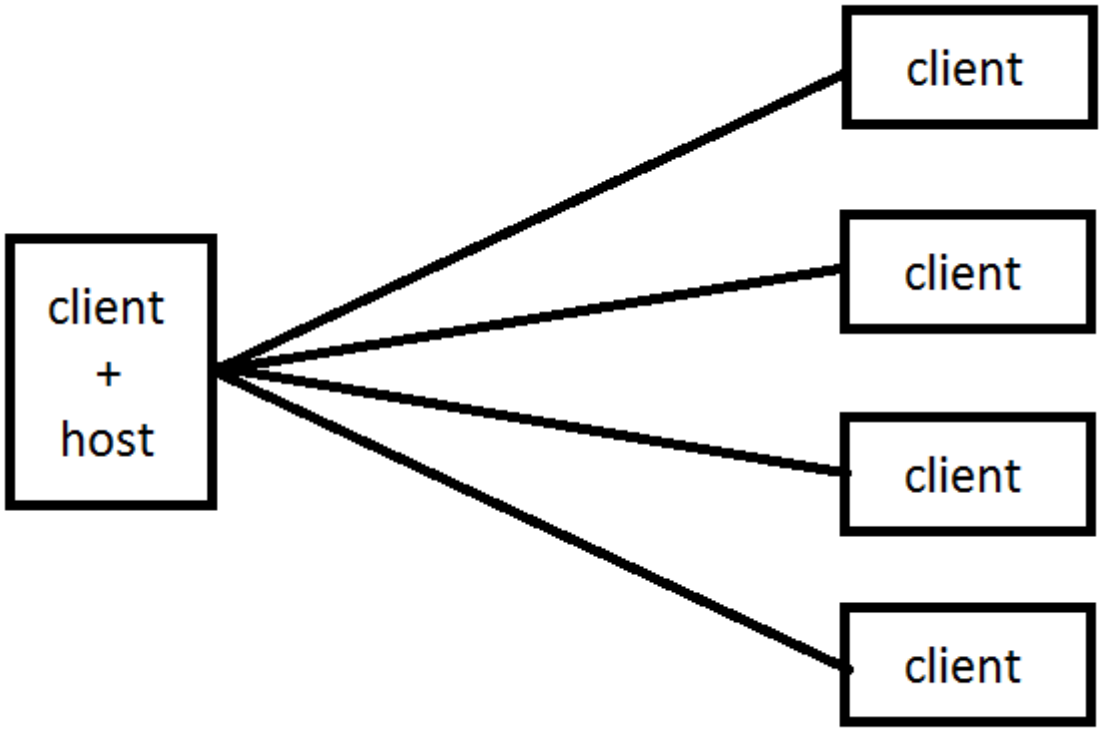

Abbildung 2.1: Das Client-Host Modell einfach veranschaulicht

Bei dieser Variante wird der Serverprozess auf dem selben Gerät ausgeführt, auf dem auch eine Client-Instanz gestartet wurde. Ein Spieler übernimmt also ßelbst"das Hosting. Andere Clients haben die Information, welcher Client einen Serverprozess "besitztünd wie sie sich dorthin verbinden können.

Vorteile: Die Unabhängigkeit von Hardwareressourcen für Entwickler, dieses "Problem"wird schlichtweg an die Spieler ausgelagert.

Nachteile: Da der Serverprozess nun ebenfalls auf einem Gerät läuft, auf welches Spieler Zugriff haben gibt es ebenfalls mehr Möglichkeiten des Hackings (Spielmanipulation). Die Entwickler haben keinerlei Einfluss auf die Serverprozesse, welche ein Spieler startet. [Sme02a]

"Client-Server":

Diese Variante trennt Client und Server physikalisch von einander. Serverprozesse werden außerhalb einer Client-Umgebung gestartet und verwaltet. Clients haben einen (oder mehrere) zentrale Zielsysteme, zu welchen sie sich verbinden können.

Vorteile:

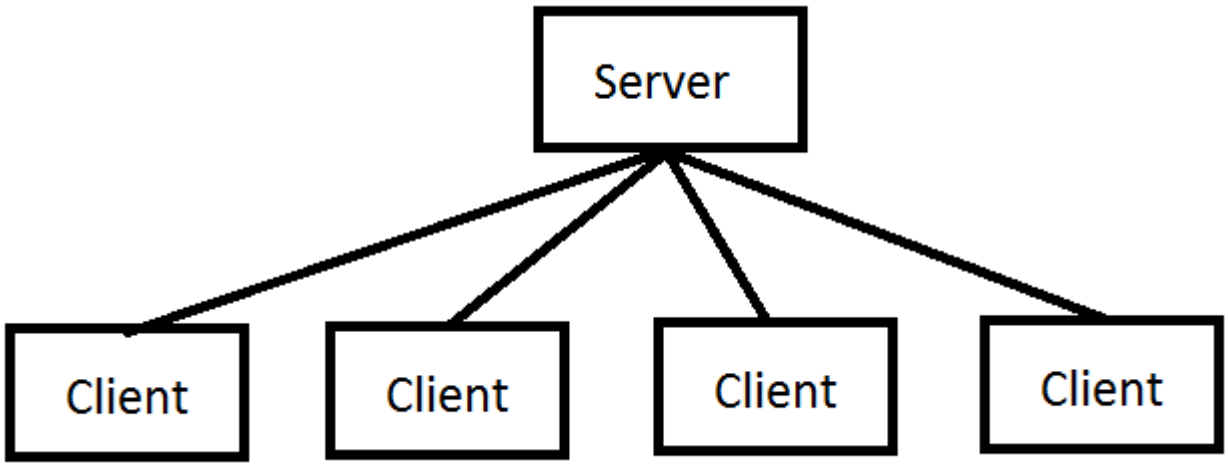

Abbildung 2.2: Das Client-Server Modell einfach veranschaulicht

Mehr "Kontrolleäuf Seiten der Entwickler. Hacking ist deutlich erschwert. Die Software-Architektur kann verhindern, dass spielentscheidende Daten nicht in der Hand der Spieler liegen, und somit ein sicheres und faires Spiel gewährleistet werden kann. [Sme02a]

Nachteile:

Die Kosten für Hardware und Bandbreite skalieren mit den Spielerzahlen. Diese Tatsache kann sehr schnell hohe Kosten verursachen. [Den18]

Ein Ausfall des zentralen Servers kann dazu führen, dass ein Großteil der Spielerfahrung nicht mehr spielbar ist.

2.3 Datenspeicherung und Transfer

Um ein bestmögliches Spielerlebnis zu ermöglichen, sollte innerhalb der Entwicklung von Online Multiplayerspielen darauf geachtet werden, dass spielentscheidende Daten schnellstmöglich zwischen den Clients synchronisiert werden. Wie auch bei anderen Echtzeitsystemen [Wik21a] kommt es also auch bei der Entwicklung von dieser Art von Software darauf an, Daten zu serialisieren [Wik19].

Daten unterscheiden sich im Kontext der Online Spielentwicklung grundsätzlich nicht großartig von anderer Software. Typischerweise werden Programmiersprachen wie C# oder C++ genutzt, um Klassen und Datenstrukturen zu erstellen, welche die eigene Spiellogik abbilden. [Gli08]

Das folgende rudimentäre Klassendiagramm zeigt zwei simple Klassen. Eine Spielerklasse, welche Informationen über den Spieler enthält sowie eine Network Manager Klasse, welche alle Spieler verwaltet.

Das Beispiel hat nicht den Anspruch auf Vollständigkeit, soll jedoch illustrieren, auf welche Art und Weise Daten typischerweise innerhalb eines Multiplayer-Kontextes verwaltet werden.

Der Entwickler entscheidet hierbei selbst, welche Informationen in welchem Kontext vorhanden ist. Konkret muss sich ein Entwickler stets die Frage stellen, ob eine Information im Server-Kontext oder im Client-Kontext verwaltet bzw. verarbeitet werden soll.

Serialisierte Daten werden über ein Netzwerk transportiert. Je Nach Art des Hostings erfolgt ein Umweg über eine Server-Instanz, oder direkt zu einem anderen Client. [Sme02b]

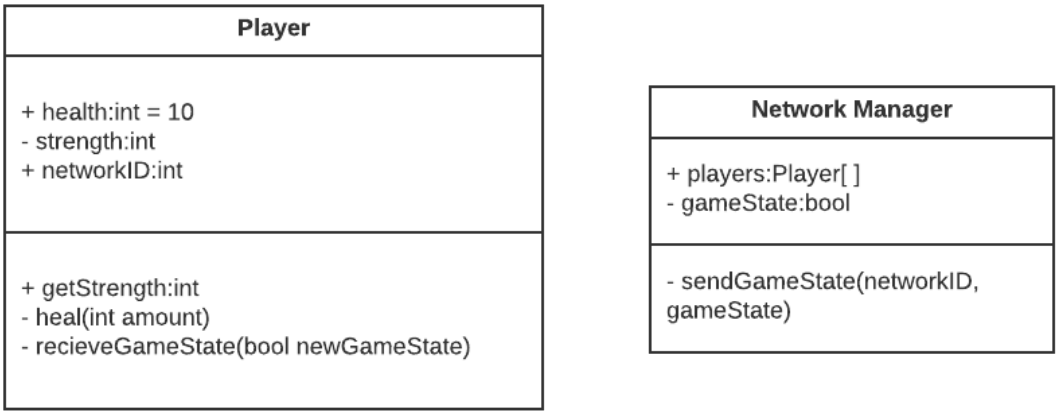

Abbildung 2.3: UML Klasse Player und Network Manager

2.4 Informationskontrolle und Interest Management

"Welche Information soll zu welchem Zeitpunkt in welchem Kontext verfügbar sein?"Diese Frage müssen sich Multiplayer-Spieleentwickler aus verschiedenen Gründen regelmäßig stellen. Die Gründe sind:

Security: –> Beispiel: Bei einem Online-Poker Spiel wäre es fatal, wenn ein Mitspieler jederzeit über alle Karten seiner Mitspieler Kenntnis hätte.

Network traffic: –> Beispiel: In Mario Kart sammelt ein Spieler ein Item ein. Der Server würfelt für diesen Spieler ein zufälliges Item aus, welches der Spieler im Anschluss verwenden soll. Die Information über das Resultat der Zufallsauswahl sollte an den jeweiligen Spieler gesendet werden, welcher dieses Item erhalten soll. Alle Anderen Spieler muss diese Information nicht interessieren.

Interest Management:

Interest Management beschreibt ein Konzept, welches deligiert, welche Informationen in welchem Kontext zu welcher Zeit für welche Personen verfügbar sind. Befindet sich der Spieler in einem bestimmten Bereich einer Spielwelt, kann das Interest Management dafür sorgen, dass dieser Spieler für manche Spieler sichtbar, und für manche Spieler nicht sichtbar ist, je nachdem wie nahe sich die Spieler zueinander befinden, oder in welchem Areal der Spielwelt sie sich befinden.

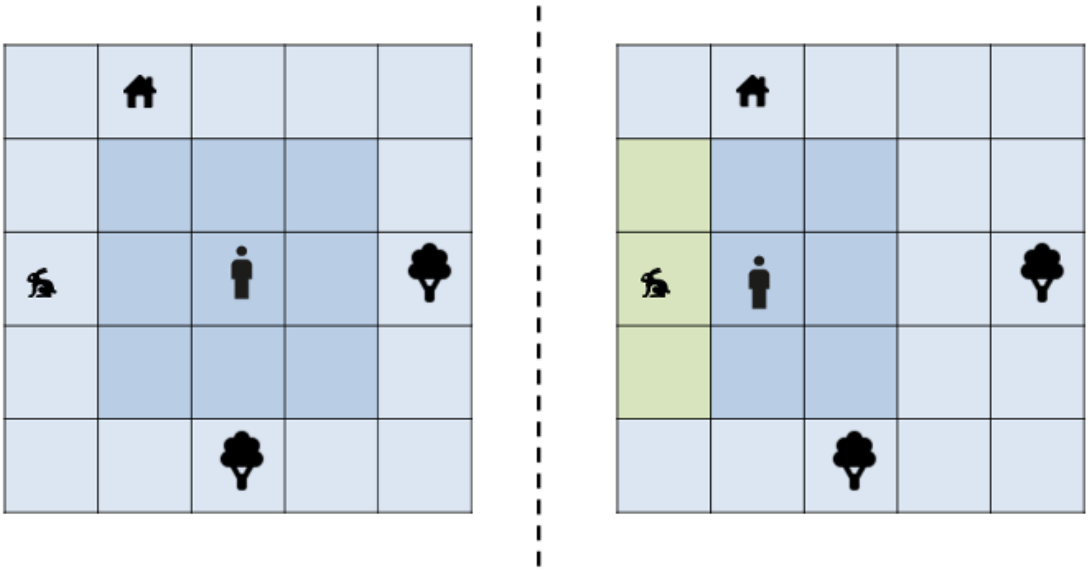

Abbildung 2.4: Illustration des Konzepts Interest Management. [Jer17]

Dieses Konzept wird besonders oft in MMO-Spielen benutzt, damit ein Server mehrere Tausend Spieler ohne Abstürze verwalten kann.

[Sme02b]

2.5 Matchmaking

Matchmaking bei Multiplayer Online Spielen ist ein Dienst, der es Spielern ermöglicht andere Mitspieler oder Gegner zu finden. [.2014]

In dieser Arbeit wird auf einige Varianten solche Matchmaking Dienste näher eingegangen. Die häufigsten Konzepte von Matchmaking Diensten sind:

Playlists: Playlists sind automatisch verwaltete Ströme von SSpielsitzungen", denen die Spieler nach Belieben beitreten und Verlassen können. Anhand einer Reihe vordefinierter Regeln wird die Konfiguration der einzelnen Sitzungen festgelegt, ohne dass ein Mensch eingreifen muss.

Die Spiele bieten in der Regel eine Auswahl an thematischen Wiedergabelisten (z. B. Teams oder Einzelspieler, ausgefallene Regeln usw.), um verschiedenen Geschmäckern oder Stimmungen gerecht zu werden. Da die Wiedergabelisten von Servern verwaltet werden, die vom Entwickler des Spiels kontrolliert werden, können sie im Laufe der Zeit geändert werden.

Wenn ein Spieler eine Wiedergabeliste auswählt, schließt er sich einem Pool von anderen Spielern an, die die gleiche Wahl getroffen haben. Der Playlist-Server verbindet sie dann entweder mit einer bestehenden Sitzung oder erstellt eine neue.

Parties:

Partys sind Gruppen von Spielern, die von Matchmaking-Systemen als eine Einheit behandelt werden. Eine Gruppe kann von Sitzung zu Sitzung wechseln, ohne dass ihre Spieler voneinander getrennt werden. Das Konzept eignet sich besonders gut für Wiedergabelisten, die automatisch die Logistik der Suche oder Erstellung von Spielsitzungen mit genügend Platz für die gesamte Gruppe übernehmen können.

Lobbies:

Lobbys sind Menübildschirme, in denen die Spieler die bevorstehende Spielsitzung einsehen, die Ergebnisse der letzten Sitzung prüfen, ihre Einstellungen ändern und miteinander sprechen können.

In vielen Spielen kehren die Spieler am Ende jeder Sitzung in die Lobby zurück. In einigen Spielen werden Spieler, die einer bereits begonnenen Sitzung beitreten, bis zum Beginn der nächsten Sitzung in der Lobby untergebracht. Da Lobbys nur wenige Ressourcen verbrauchen, werden sie manchmal zusätzlich als "Warteschleife"für Spieler verwendet, bis ein geeigneter Gastgeber für die nächste Sitzung gefunden ist.

Lobbys, die von Playlists erstellt werden, haben oft einen Countdown-Timer, bevor die Sitzung beginnt, während Lobbys, die von einem Spieler erstellt werden, im Allgemeinen nach dessen Ermessen übergehen.

Ranking:

Viele Matchmaking-Systeme verfügen über ein Ranglistensystem, mit dem versucht wird, Spieler mit ungefähr gleichen Fähigkeiten zusammenzubringen. Ein Beispiel dafür ist das TrueSkill-System von Xbox Live.

Spiele wie League of Legends verwenden Divisionen und Ränge für ihr Matchmaking-Bewertungssystem. Jeder Spieler tritt in verschiedenen Rängen an: Eisen, Bronze, Silber, Gold, Platin, Diamant, Meister, Großmeister, Herausforderer, und für jeden Sieg gibt es Ligapunkte, für jede Niederlage werden Ligapunkte abgezogen.

Bei Spielen mit Rangliste werden in der Regel nicht gewertete Sitzungen für Spieler angeboten, die nicht wollen, dass ihre Leistung aufgezeichnet und analysiert wird. Diese werden getrennt gehalten, damit sich rangierte und nicht rangierte Spieler nicht vermischen. Videospiele wie League of Legends, Counter-Strike: Global Offensive, Rocket League und Brawlhalla verwenden das Elo-Bewertungssystem in ihren Ranglistenspielen.

Server browsers:

Einige Spiele (vor allem solche mit dedizierten Servern) zeigen den Spielern eine Liste aktiver Sitzungen an und ermöglichen ihnen, manuell eine Sitzung auszuwählen. Dieses System kann in Verbindung mit Ranglisten und Lobbys verwendet werden, wird aber durch die On-Demand-Sitzungserstellung von Wiedergabelisten vereitelt.

Die meisten dieser Server-Browser ermöglichen es den Spielern, die Ergebnisse zu filtern, die sie liefern. Zu den üblichen Filterkriterien gehören Servername, Spielerzahl, Spielmodus und Latenzzeit.

Contacts lists:

Eine der grundlegendsten und gängigsten Formen des Matchmaking besteht darin, den Spielern eine Liste anderer Spieler zur Verfügung zu stellen, die sie bereits kennengelernt haben und mit denen sie vielleicht wieder spielen möchten. Der Status jedes Spielers

(offline, online, spielend) wird angezeigt, es besteht die Möglichkeit, einer laufenden Sitzung beizutreten, und im Allgemeinen ist es möglich, Chat-Nachrichten zu senden.

In vielen Fällen werden die Kontaktlisten von der Plattform verwaltet, auf der ein Spiel läuft (z. B. Xbox Live, PlayStation Network, Steam), um den Spielern den Aufwand zu ersparen, viele separate Listen für viele einzelne Spiele zu verwalten.

[Wik21c]

// WEITERE IDEEN: Consistency and Responsiveness

3 Konzepte

In diesem Kapitel werden Konzepte beschrieben, die möglichst abstrakt gehalten sind. Entwickler eines Online-Multiplayer Spiels sollen anhand von diesen Konzepten eine Hilfestellung bekommen, um einen roten Faden in ihrer Entwicklung verfolgen zu können.

Bevor eines dieser Konzepte angewandt werden kann, müssen folgende Voraussetzungen erfüllt sein:

1. Das grundsätzliche Spielkonzept wurde erarbeitet und steht fest.

2. Anhand von dem festgelegten Spielkonzept kann abgeschätzt werden, ob sich für das Spiel eine Client / Server oder eine Client / Host Architektur eignet.

Sollte das Team bzw. der Solo-Entwickler zum Entschluss gekommen sein dass sich ein Client / Server Modell am besten für den spezifischen Use-Case eignet, so müssen vorab Vorkehrungen getroffen werden, um die Hardware-Ressourcen für den späteren Server-Runner sicher zu stellen. Die Grundvoraussetzung ist jedoch ein aus dem Internet erreichbarer PC mit einer festen IP-Adresse.

> Distribution concepts
>
> "There is not much we can do about the aforementioned resource limitations. Therefore, the problems of distributed interactive systems should be tackled on a higher level, which means choosing architectures for communication, data, and control. Still, sometimes the architecture alone cannot rid the system of resource limitations, and compensatory techniques are needed to relax the requirements."[Sme02a]

Nun gilt es einen passenden Technologie-Stack zu finden. Hier gibt es keine Klare Empfehlung, jedoch können die in diesem Kapitel beschriebenen Konzepte als Entscheidungshilfe dienen. Auf dem offenen Markt existieren bereits Frameworks & Game Engines, die Manche dieser Konzepte bereits implementiert haben und als Libraries für Entwickler bereitstellen.

[MFa21]

3.1 API für Matchmaking & Server Runner

Bei diesem Konzept muss eine API entworfen werden, welche unabhängig von einem existierenden Client oder Serverprozess arbeitet. Diese API hat 2 grundsätzliche Aufgaben.

Matchmaking:

Die API muss einen Algorithmus implementieren, welcher mehrere Spieler zu einer Spiel-Session zusammenführt. Mögliche Matchmaking Konzepte sind bereits im Hintergrund-Kapitel beschrieben. Die Matchnaking API verwaltet als eine Liste an aktiven Serverprozessen sowie die Information, wie man sich zu ihnen verbindet. In der Regel wird pro Serverprozess ein Netzwerk-Port an der Server-Maschine reserviert, auf denen sich dann N Spieler verbinden können.

Je nach Spielkonzept können hunderte, tausende Serverprozesse existieren, welche jeweils nur eine vergleichsweise geringe Anzahl (1-20) an Spieler verwalten. Spielkonzepte, welche viele Spieler (200-1000) innerhalb eines einzigen Serverprozesses voraussetzten, erzeugen dagegen zwar quantitativ weniger Serverprozesse, diese neigen aber in der Regel schnell zu Überlastung. Neben der Umsetzung von Interest Management und einer schlanken Architektur für möglichst wenig Network-Traffic kann aber auch die Matchmaking API Abhilfe schaffen, bspw. durch Ümverlegung"von Spielern auf andere oder neue Serverprozesse.

Server Runner:

Neben dem Matchmaking ist die API ebenfalls auch zuständig für das Starten und die Überwachung von Serverprozessen. Ebenfalls können technische Statusinformationen über aktuell laufende Spiel innerhalb eines Serverprozesses von der API verwaltet werden (Bspw. ob es möglich ist einem Server beizutreten, oder wieviele Spieler bereits auf einem Serverprozess spielen)

Je nach Use-Case kann das Aufgabenfeld der API um weitere Punkte erweitert werden, hier ein paar Beispiele: –> Datenbank für Authentifizierung bzw. Kommunikation mit externen SSO-Service Providern [Wik21f] –> Datenbank für InGame Shop / Kommunikation mit externen Lösungen

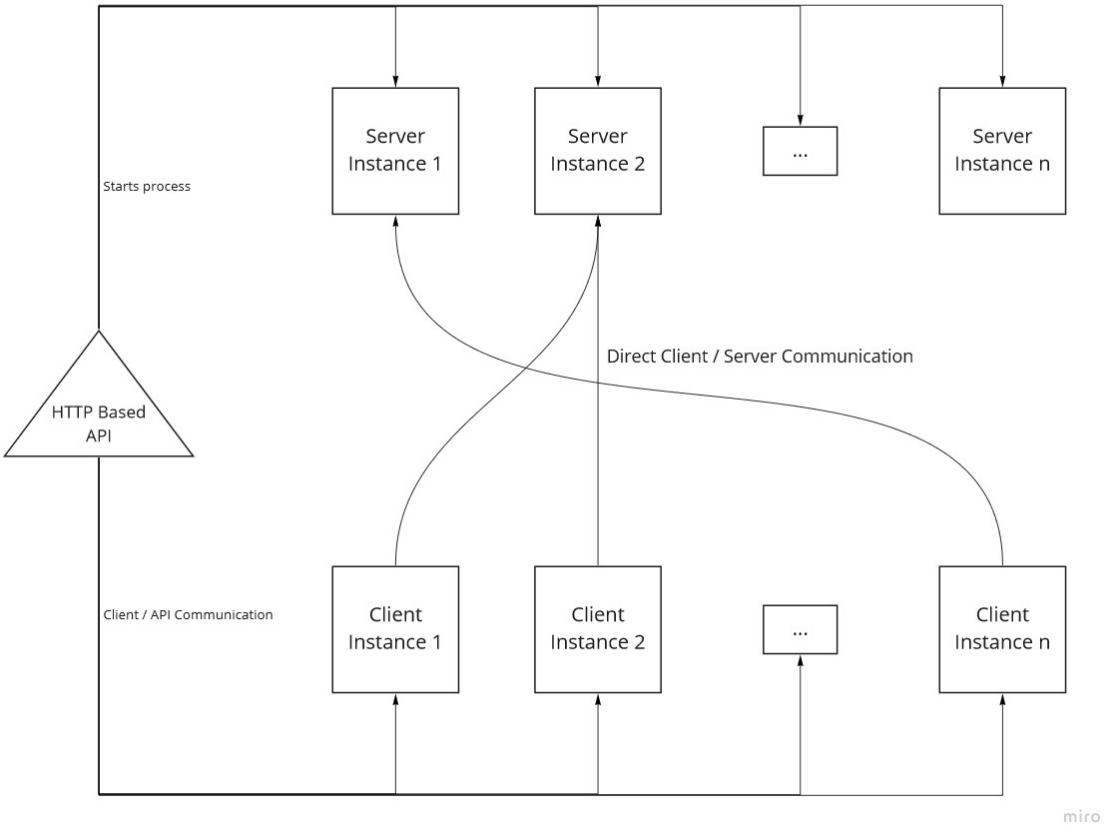

Abbildung 3.1: Veranschaulichung des API Konzepts

3.2 Client UI & Visual Controller

Das Konzepts des Client UI & Visual Controllers beschreibt die Art und Weise, wie die Benutzeroberfläche und visuelle Ebene eines Spielers durch Antworten des Servers oder einzelner Spieleraktivitäten manipuliert wird.

Konkret sorgt ein Server-Ereignis, der Input eines Spielers (beispielsweise durch klicken eines Buttons oder einsammeln eines Gegenstands) oder eine andere Bedingung dafür, dass Funktionen innerhalb des Client UI & Visual Controllers aufgerufen werden. Innerhalb dieser Funktionen werden Komponenten der Benutzeroberfläche wie z.b. Anzeigetexte, Bilder etc. oder Objekte innerhalb der Spielwelt selbst manipuliert.

Wichtig ist, dass Funktionen des Client UI & Visual Controller erst nach server- oder clientseitigen Überprüfungen ausgeführt werden. Der Controller selbst sollte nur überprüfen, ob die Komponente, welche er manipulieren will überhaupt existiert. Spielrelevante Logik, beispielsweise ob ein Spieler berechtigt ist, bestimmte Aktionen durchzuführen, ist nicht Teil des Client UI & Visual Controllers.

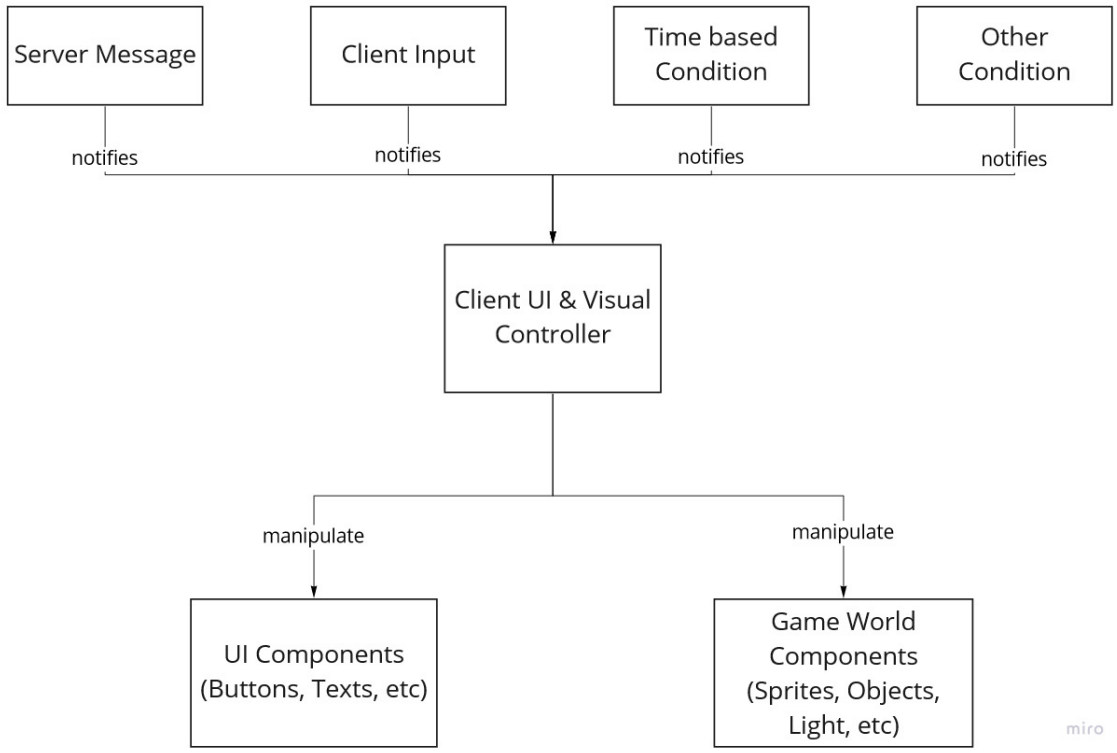

Abbildung 3.2: Veranschaulichung des Client UI & Visual Controller Konzepts

3.3 Client Connection Manager

Das Konzept Client Connection Manager beschreibt eine Softwarekomponente, welche folgende Aufgaben umsetzen muss:

- Implementierung von clientseitigen Methoden zum Informationsaustausch mit Matchmaking API. Z.B. stellt die Matchmaking API dem Client über HTTP-Kommunikation Informationen bereit, mit denen sich der Client auf einen speziellen Server verbinden kann (IP + Port). [Wik21b] [Wik21e]

- Verarbeitung der von Matchmaking API bereitgestellten Informationen.

Z.B. könnte die Matchmaking API einem Client die Information bereit stellen, dass aus einem bestimmten Grund dem angeforderten Serverbeitritt verweigert wird. Diese Information muss der Client Connection Manager SZwischenspeichernünd verwalten.

- Aufruf von Methoden eines Client UI Controllers.

Z.B. Die soeben erlangte Information über den verweigerten Serverbeitritt muss nun dem Nutzer dargestellt werden, hierfür ruft der Client Connection Manager Funktionen auf, welche ein Client UI Controller implementiert hat.)

- Ausführen von Handler-Funktionen und Auslösen von Callbacks für clientseitige Konsequenzen bei Netzwerkabbruch, manuellem Verlassen einer Server-Session oder sonstigen Abbruchgründen, welche dazu führen, dass eine Online-Spielsession verlassen wird.

Z.B. Wird die notwendige Verbindung zum Internet während des Spiels unterbrochen. Der Client Connection Manager löst Callbacks aus, welche im Client UI & Visual Controller implementiert sind. Diese Sorgen dafür, dass der Spieler zurück ins Hauptmenü geleitet wird, in dem eine Fehlermeldung dargestellt wird, welche erklärt warum das Spiel abgebrochen ist.

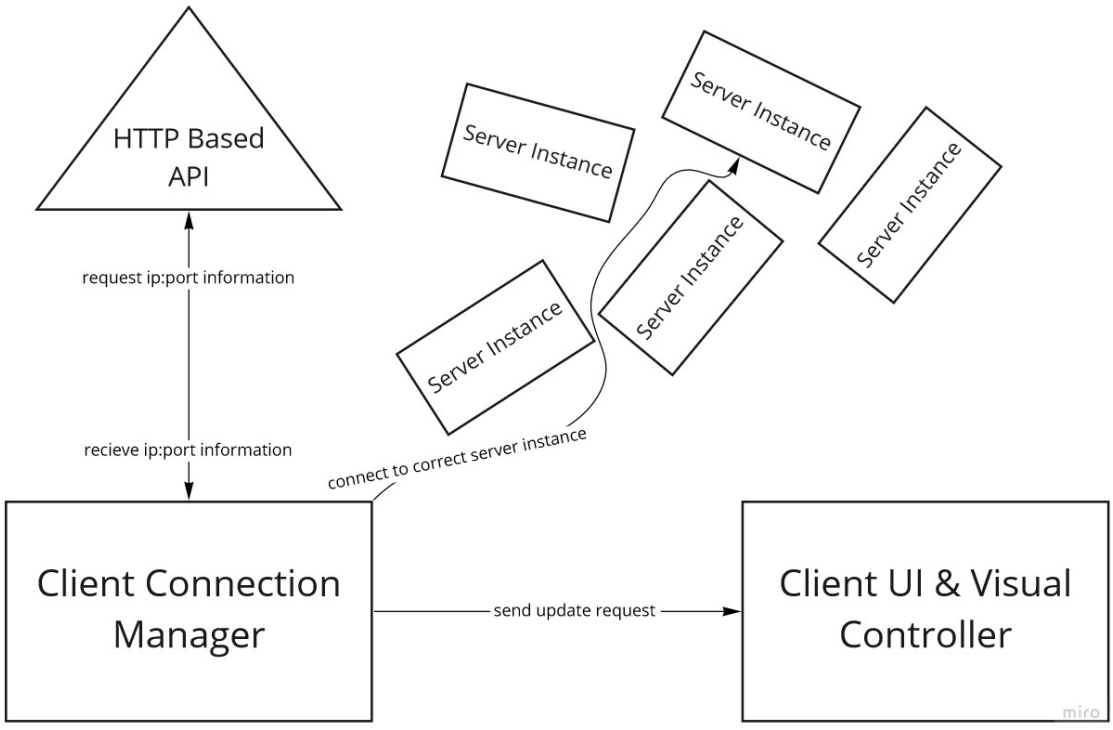

Abbildung 3.3: Veranschaulichung des Client Connection Manager Konzepts

3.4 Serverside Client Manager

Das Konzept des serverseitigen Client Manager beinhaltet die Verwaltung aller verbundener Clients innerhalb eines aktiven Serverprozesses. Insbesondere speichert der serverseitige Client Manager 2 Arten von Information über jeden Client:

Einerseits werden Netzwerk-bezogene Informationen über einen Client gespeichert, im einfachsten Fall ist dies lediglich die IP Adresse des Hosts sowie der Port, auf welchem der Client seinen Kommunikationskanal geöffnet hat. Bei Spielkonzepten, welche eine weitergehende Authentifizierung des Spielers voraussetzen, können hier auch Benutzernamen, Benutzertags, Email-Adressen o.ä. gespeichert und weiterverarbeitet werden.

Zum Anderen speichert und verwaltet der serverseitige Client Manager auch spielspezifische Informationen wie Bspw. den aktuellen Anzeigenamen, konfigurierte Individualisierung des Spielcharakters, Spielerrollen oder andere Eigenschaften, über die der Server während einer Spiel-Session Kenntnis haben sollte.

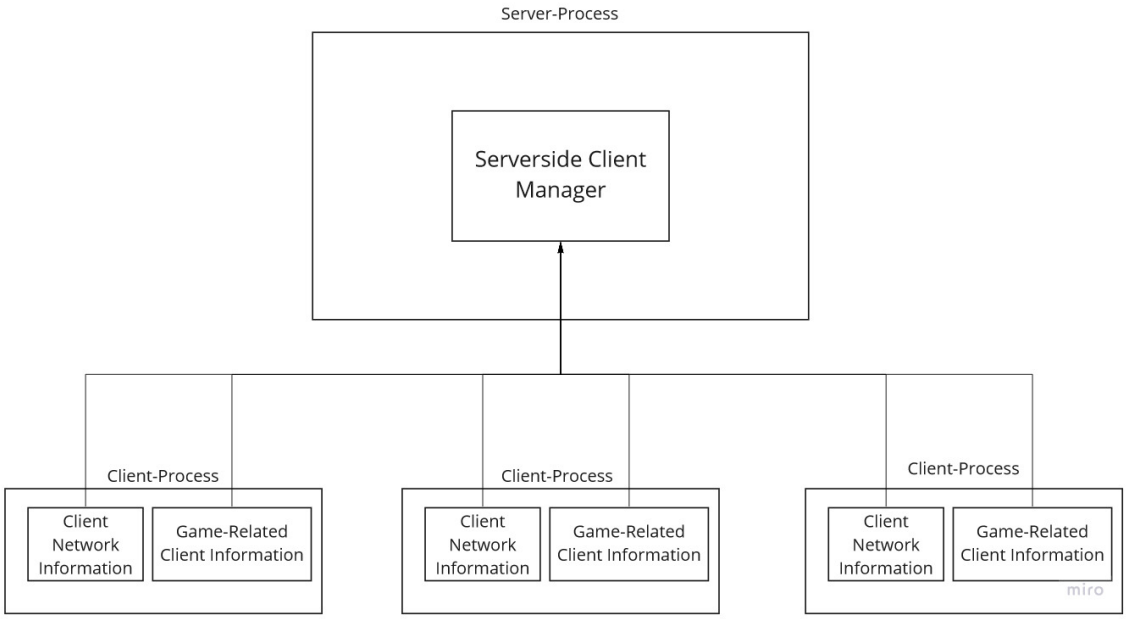

Abbildung 3.4: Veranschaulichung des serverseitigen Client Manager Konzepts

3.5 Prepare-Game-Manager

Der Prepare-Game-Manager ist Teil des Serverprozesses und beinhaltet alle Funktionen, welche nötig sind um eine Spielszene vorzubereiten, bevor die Spieler ihr beitreten dürfen. Die Funktionen sollten konkret umsetzen:

- Spawning [Wik20] von NPCs [Wik21d] oder Spielgegenständen, die in Abhängigkeit zur Anzahl der beitretenden Spieler, einer Spielkonfiguration oder sonstigen Parametern stehen.

- Anpassung der Eigenschaften von bereits gespawnten NPCs oder Spielgegenständen, die in Abhängigkeit zur Anzahl der beitretenden Spieler, einer Spielkonfiguration oder sonstigen Parametern stehen.

- Initialisierung der Spawnpunkte für Spieler anhand von Spawnalgorithmen oder festgelegten Punkten in der Spielwelt.

- Abarbeitung sonstiger serverseitigen Abläufe, welche Voraussetzung für den Start des Spiels sind

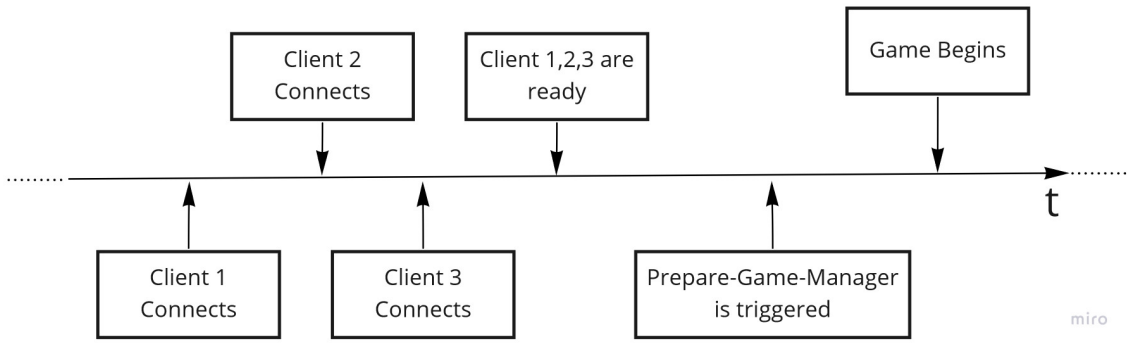

Abbildung 3.5: Veranschaulichung des Prepare Game Manager Konzepts

3.6 Progress / Game-State Manager

Die Aufgabe eines Progress bzw. Game-State Managers ist es, Gewinn bzw. Verlustbedingungen einer Spielsession zu speichern und zu verwalten.

Beispiele hierfür sind: - Verwaltung und Synchronisierung von globalen Timern - Verwaltung und Synchronisierung von Spiel-Kennzahlen (z.B. der Spielstand bei einem Fußball-Match)

Außerdem ist der Progress Manager / Game-State Manager dafür verantwortlich, alle Clients über spielentscheidende Ereignisse zu informieren.

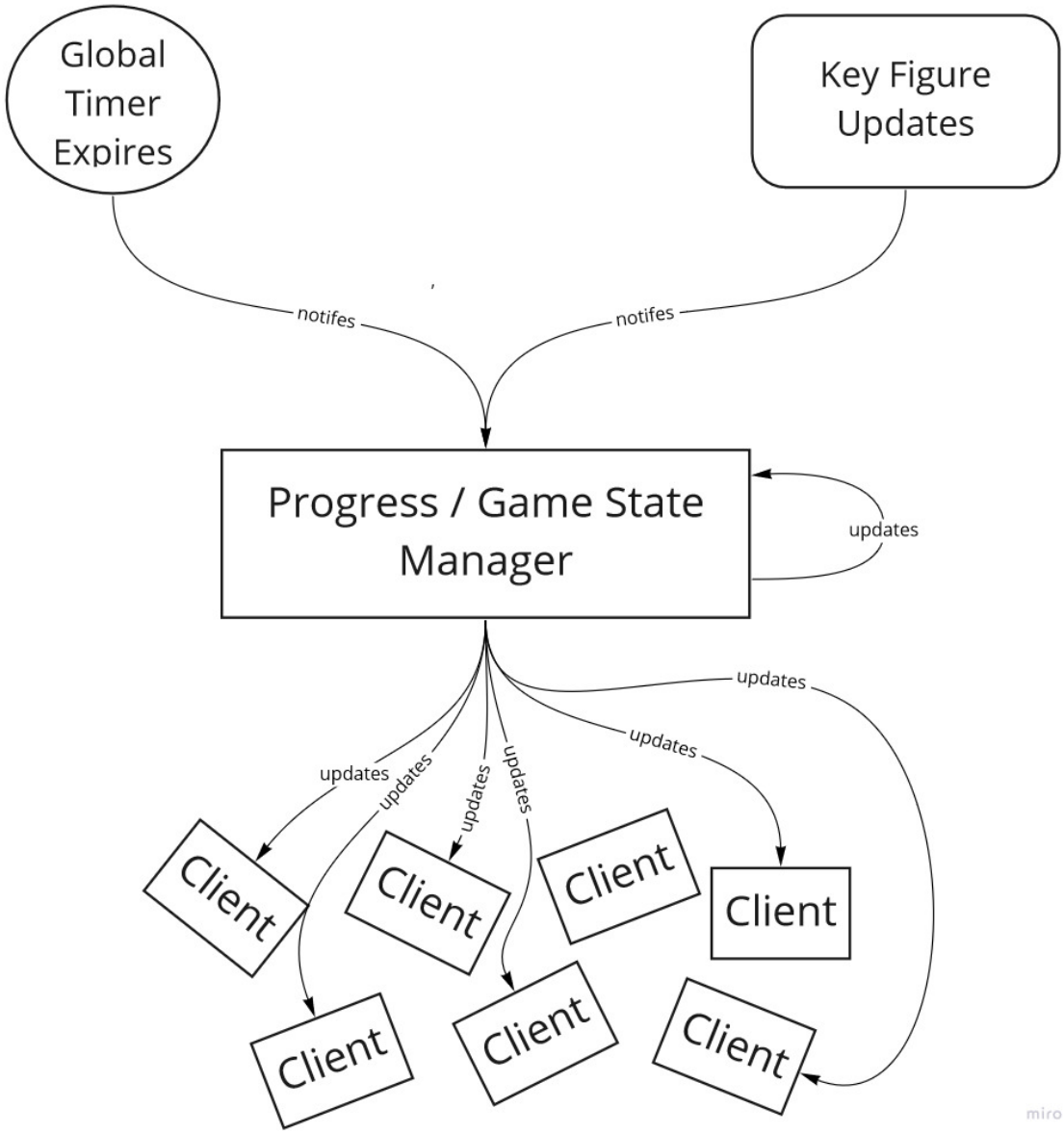

Abbildung 3.6: Veranschaulichung des Progress / Game-State Manager Konzepts

3.7 Spawn Manager

// TODO Add Content

3.8 Entwicklung sonstiger Server Logik

(Hier vielleicht noch ein paar Konzepte rausarbeiten)

3.9 Entwicklung sonstiger Client Logik

(Hier vielleicht noch ein paar Konzepte rausarbeiten)

3.10 Studienkonzept

// TODO Add Content

Wie können diese Konzepte auf Hand und Fuß überprüft werden?

4 Realisierung

5 Zusammenfassung

5.1 Fazit

5.2 Auswertung

5.3 Weitere Ansätze

5.4 Nächste Schritte

5.5 Ausblick

Literaturverzeichnis

[.2014] *Matchmaking in multi-player on-line games: studying user traces to improve the user experience* (2014)

[Den18] Deng, Yunhua; Li, Yusen; Seet, Ronald; Tang, Xueyan und Cai, Wentong: The Server Allocation Problem for Session-Based Multiplayer Cloud Gaming. *IEEE Transactions on Multimedia* (2018), Bd. 20(5): S. 1233–1245

[gam20] game - Verband der deutschen Games-Branche e.V.: Jahresreport der deutschen Games-Branche 2020 (2020), URL https://www.game.de/wp-content/uploads/2020/08/game-Jahresreport-2020.pdf

[Git21] GitHyp: Most played games on Steam in 2020, by hourly average number of players (February 2021), URL https://www.statista.com/statistics/656319/steam-most-played-games-average-player-per-hour/

[Gli08] Glinka, Frank; Ploss, Alexander; Gorlatch, Sergei und Müller-Iden, Jens: High-Level Development of Multiserver Online Games. *International Journal of Computer Games Technology* (2008), Bd. 2008: S. 1–16, URL https://www.hindawi.com/journals/ijcgt/2008/327387/

[Jer17] Jerome Renaux: Interest management for multiplayer online games (2017), URL https://www.dynetisgames.com/2017/04/05/interest-management-mog/

[MFa21] MFatihMAR: Game Networking Resources (2021)

[Pay19] Payne, John: Making Multiplayer Games Doesn't Have to Be Difficult. *Crayta* (18.09.2019), URL https://medium.com/crayta/making-multiplayer-games-doesnt-have-to-be-difficult-d08d19c83de

[Sim20] Simon-Kucher & Partners: Impact of COVID-19 on the frequency of playing multiplayer video games worldwide as of June 2020 (2020), URL https://www.statista.com/statistics/1188549/covid-gaming-multiplayer/

[Sme02a] Smed, Jouni; Kaukoranta, Timo und Hakonen, Harri: Aspects of networking in multiplayer computer games. *The Electronic Library* (2002), Bd. 20(2): S. 87–97, URL https://www.emerald.com/insight/content/doi/10.1108/02640470210424392/full/pdf

[Sme02b] Smed, Jouni; Kaukoranta, Timo und Hakonen, Harri: *A review on networking and multiplayer computer games*, Citeseer (2002), URL

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.2565&rep=rep1&type=pdf

[Wik19] Serialisierung (2019), URL https://de.wikipedia.org/w/index.php?title=Serialisierung&oldid=190015284

[Wik20] Spawnen (2020), URL https://de.wikipedia.org/w/index.php?title=Spawnen&oldid=200710033

[Wik21a] Echtzeitsystem (2021), URL https://de.wikipedia.org/w/index.php?title=Echtzeitsystem&oldid=217468472

[Wik21b] IP-Adresse (2021), URL https://de.wikipedia.org/w/index.php?title=IP-Adresse&oldid=218224804

[Wik21c] Matchmaking (video games) (2021), URL https://en.wikipedia.org/w/index.php?title=Matchmaking_(video_games)&oldid=1056891415

[Wik21d] Nicht-Spieler-Charakter (2021), URL https://de.wikipedia.org/w/index.php?title=Nicht-Spieler-Charakter&oldid=217227874

[Wik21e] Port (Protokoll) (2021), URL https://de.wikipedia.org/w/index.php?title=Port_(Protokoll)&oldid=216781784

[Wik21f] Single Sign-on (2021), URL https://de.wikipedia.org/w/index.php?title=Single_Sign-on&oldid=215677155

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

.

A Anhang 1

B Anhang 2