**Christoph-Scheiner-Gymnasium**

**Ingolstadt**

Seminararbeit

aus dem

wissenschaftspropädeutischen Seminar

im Fach Informatik

**Implementierung des Spiels „ХАЛЁ“ als Browser-Spiel mit Augenmerk auf die Server-Client Kommunikation**

|  |  |
| --- | --- |
| Angefertigt von | Adrian Rarov |
| Reifeprüfungsjahrgang | 2022 |
| Kursleiter | StD Martin Pabst |
|  |  |

Inhaltsverzeichnis

[1 Einleitung: Beschreibung von ХАЛЁ 4](#_Toc85914689)

[1.1 Spielidee 4](#_Toc85914690)

[1.2 Spielprinzip 4](#_Toc85914691)

[1.3 Spielaufbau 4](#_Toc85914692)

[2 Die Server-Client Kommunikation 5](#_Toc85914693)

[2.1 Die Namenseingabe des Clients 5](#_Toc85914694)

[2.1.1 DOM 5](#_Toc85914695)

[2.2 Websockets 5](#_Toc85914696)

[2.2.1 Was ist ein Websocket? 5](#_Toc85914697)

[2.2.2 Warum Websockets für ХАЛЁ? 5](#_Toc85914698)

[2.2.3 Express-Server 5](#_Toc85914699)

[2.2.4 Messages 5](#_Toc85914700)

[2.2.5 Der WebsocketController 5](#_Toc85914701)

[2.2.6 Der Server 5](#_Toc85914702)

[2.2.7 Der Client 5](#_Toc85914703)

[2.3 Die Darstellung einer Runden-Lobby 5](#_Toc85914704)

[2.3.1 Das Boxes-System 5](#_Toc85914705)

[2.3.2 Das Code-DOM 5](#_Toc85914706)

[3 Die Realisierung der Design-Vision 6](#_Toc85914707)

[3.1 Die Konzeptzeichnungen 6](#_Toc85914708)

[3.2 Die 4-Zeichen Struktur 6](#_Toc85914709)

[3.3 Random Favicons 6](#_Toc85914710)

[3.4 Webfonts und CSS 6](#_Toc85914711)

[4 GameDev zu zweit 6](#_Toc85914712)

[4.1 git 6](#_Toc85914713)

[5 Ausblick in die Zukunft 7](#_Toc85914714)

[5.1 Verbesserungs- und Erweiterungsmöglichkeiten 7](#_Toc85914715)

[5.1.1 Teams 7](#_Toc85914716)

[5.1.2 Power-Ups 7](#_Toc85914717)

[5.1.3 Angriffe 7](#_Toc85914718)

[5.2 Monetarisierungsmöglichkeiten 7](#_Toc85914719)

[6 Fazit/Schlusswort 8](#_Toc85914720)

[7 Literaturverzeichnis 9](#_Toc85914721)

[8 Eidesstattliche Erklärung 10](#_Toc85914722)

# Einleitung: Beschreibung von ХАЛЁ

## Spielidee

Leo und ich hatten mehrere Ideen welches Spiel wir umsetzen könnten, und haben uns schlussendlich für eine Art Remake eines absoluten Klassikers entschieden: Tetris. Die Idee war grundsätzlich Tetris einen Multiplayer-Twist zu geben. Da haben wir viel nachgedacht und mögliche Konzepte überlegt, und haben uns auf das Spielprinzip geeinigt, bis zu 5 Spieler gegenseitig die ausgefüllten Reihen per Drag and Drop zu schieben zu lassen.

## Spielprinzip

Die

## Spielaufbau

Text…

# Die Server-Client Kommunikation

## Die Namenseingabe des Clients

### DOM

## Websockets

### Kommunikation per HTTP

Browser kommunizieren normalerweise mit Servern per http. Wenn der Benutzer eine URL oben in die Adresszeile eingibt, schickt der Browser eine http Request an den entsprechenden Web-Server. Dieser schickt eine http Response zurück. Im http Protokoll liegt eine TCP Verbindung zugrunde. Diese wird vor dem http Request geöffnet und nach der http Response wieder geschlossen. Danach wissen Client und Server gar nichts mehr voneinander, es sei denn man greift auf zusätzliche Techniken wie Cookies zurück.

### Warum Websockets?

Nur der Client kann eine http-Kommunikation beginnen.  Denn nach der Response des Servers wird die TCP-Verbindung wieder abgebaut und der Server kann von sich aus den Client gar nicht erreichen. Gibt es jetzt beim Server Ereignisse, die er dem Client mitteilen will, zum Beispiel, weil bei einem Netzwerkspiel irgendein anderer Spieler seinen Charakter bewegt hat oder einen Schuss abgefeuert hat, dann muss der Server zunächst mal warten bis der Client das nächste Mal anfangt. Erst dann kann er ihm sagen: „hey da ist was passiert“. Für Spiele geht das natürlich überhaupt nicht. Es gibt Krücken, um damit umzugehen, zum Beispiel: Polling. Polling bedeutet, der Client fragt zum Beispiel alle Viertelsekunden beim Server an: "Hey, gibt's was Neues? Gibt's was Neues?" und dann wenn der Server irgendwas Neues hat, schickt er es ihm. Aber das ist nicht sehr performant und man kann die Frequenz auch nicht zu hochtreiben. Ja, eine Viertelsekunde geht vielleicht doch einigermaßen, aber selbst das belastet die Verbindung schon sehr stark, aber stellt euch mal vor ihr habt ein Netzwerkspiel und zwischen einem neuen Ereignis, weil irgendein Spieler seinen Charakter bewegt und dem Mitteilen des Servers an die anderen Spieler liegt immer mindestens eine Viertelsekunde. Das geht nicht, das ist viel zu lang. Ein weiterer Grund warum http nicht sehr performant abläuft ist, dass bei jeder http Request wieder eine neue TCP-Verbindung geöffnet und nach der Request geschlossen wird. Es gibt zwar inzwischen Techniken, um diese TCP-Verbindungen wiederzuverwenden, wenn sie schon mal aufgebaut sind, aber das ist ein anderes Thema. Es funktioniert alles relativ schlecht. Deshalb haben sich die Verantwortlichen bei den Standardisierungsgremien schon vor ein paar Jahren zusammengesetzt und gesagt: "Warum ermöglicht man es dem Browser nicht dauerhaft TCP-Verbindungen mit Servern aufzubauen und aufrechtzuerhalten?" Und über diese könnte man dann bidirektional Daten schicken. Genau das ermöglichen Websockets.

### Kommunikation per WebSockets

Will ein Client eine WebSocket-Verbindung mit einem Server beginnen, so schickt er zunächst mal einen http Upgrade-Request. Das ist eine ganz normale http Request, in der ein bestimmtes Schlüsselwort untergebracht wird: Upgrade. Ist der Server für Websocket-Verbindungen eingerichtet, dann antwortet er „Upgrade ok“ und ab diesem Zeitpunkt lässt der Server die TCP-Verbindungen offen und diese TCP Verbindung kann bidirektional genutzt werden.

### Express-Server

### Messages

### Der WebsocketController

### Der Server

#### Der MainServer

#### Die Berechnung und Struktur des Spiels

### Der Client

#### Die Lobby-Scene

#### Die Darstellung des Spiels

## Die Darstellung einer Runden-Lobby

### Das Boxes-System

### Das Code-DOM

# Die Realisierung der Design-Vision

## Die Konzeptzeichnungen

Text…[[1]](#footnote-1) und noch ein Text[[2]](#footnote-2)

Und zudem steht fest, dass „[der Gerthsen für die] Studierenden der Physik im Haupt- und Nebenfach […] ein unverzichtbarer und kompetenter Begleiter durch das gesamte Studium [ist]“[[3]](#footnote-3)

## Die 4-Zeichen Struktur

## Random Favicons

## Webfonts und CSS

# GameDev zu zweit

## git

# Ausblick in die Zukunft

## Verbesserungs- und Erweiterungsmöglichkeiten

### Teams

### Power-Ups

### Angriffe

## Monetarisierungsmöglichkeiten

# Fazit/Schlusswort

Text…

# Literaturverzeichnis

*1.1 Webserver im 15. Jahrhundert*

https://www.gibtsgarnicht.de/ertserOrdner/Seite.html

aufgerufen am 05.11.2016

*1.2 Webserver im 15. Jahrhundert*

https://www.gibtsgarnicht.de/ertserOrdner/Seite.html

aufgerufen am 05.11.2016

*1.2 Gerthsen Physik (Springer-Lehrbuch)*

Dieter Meschede

Springer-Verlag 20152

# Eidesstattliche Erklärung

Ich erkläre, dass ich die Seminararbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis aufgeführten Quellen und Hilfsmittel verwendet habe.

Ingolstadt, den dd.mm.yyyy

1. Siehe Quelle 1.1, Abschnitt „Performance“ [↑](#footnote-ref-1)
2. Siehe Quelle 1.2, Seite 325, f. [↑](#footnote-ref-2)
3. Siehe Quelle 1.3 [↑](#footnote-ref-3)