

TODO

■ Formulierung und Quelle	2
■	3
■	3
■ Im Netzwerk! Im Gegensatz zum lokalen Multiplayer.	8
■ Welche Eigenschaften/Merkmale hat der lokale Multiplayer, die auch im Netzwerk-Multiplayer gelten sollen	8
■ Evtl. ganz an den Anfang schieben, damit man im Multiplayer-Abschnitt darauf verweisen kann.	9
■ Hier nur beschreiben. Hier kann ich letztlich auch nur raten, dass die gewählten Technologien sinnvoll sein werden. Ob die getroffene Auswahl gut war (wurden Ziele erreicht?), zeichnet sich ggf. schon während der Implementierung ab. Ergebnisse und Erkenntnisse, die die Bildung eines ersten Urteils ermöglichen, werden erst in der Evaluationsphase geschaffen.	10
■ NICHT: Vergleichen/Abwägen, Entscheiden, Begründen	10
■ Kurze Einordnung	12
■ Völlig neu? Alt?	12
■ Einfache Lösung? Total schwer?	12
■ Forschung? Anwendung?	12
■	12

■ Passt wahrscheinlich besser woanders hin	19
■	19
■ https://gamedev.stackexchange.com/questions/74973/maximum-audio-delay-before-the-player-notices	22
■ Glossar-Links	23
■ warum erfüllen die beiden das? Kompatibilitätsmatrizen, iswebrtcreadyyet	23
■ Browser-Liste, die die Anfordungen im Moment erfüllen?	23
■ Meine eigenen, evtl. vorher entwickelten. Third-Party natürlich.	24
■	57
■	57
■	57
■	59

Masterarbeit
Entwicklung einer auf Web-Standards basierenden
Multiplayer-Online-Plattform für emulierte
(S)NES-Konsolenspiele

Kai Kühne
798797

Beuth Hochschule für Technik Berlin
Fachbereich VI – Informatik und Medien

Berlin, 27. Juli 2017

Betreuung: Herr Dipl.-Inform. Hans-Georg Reimer (LB)
Begutachtung: Herr Prof. Dr. Hildebrand

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	3
1.2 Zielsetzung	4
1.3 Vorgehensweise	5
2 Grundlagen	7
2.1 Emulation von Konsolenspielen	8
2.2 Der Multiplayer-Modus	8
2.3 Grundbegriffe der Netzwerk-Kommunikation	9
2.4 Das Realtime-Web	10
3 Spezifikation	11
3.1 Funktionale Anforderungen	14
3.2 Nichtfunktionale Anforderungen	21
4 Probleme & Lösungsansätze	25
4.1 Web-Applikation in Echtzeit	26
4.2 Emulation im Web	26

4.3	Transport & Synchronisation der Spieldaten	26
4.4	Kommunikation zwischen Browser und Controller-App	26
5	Lösungsansätze	27
5.1	Vorhandene Teillösungen	28
5.2	Lösungsansatz 1: Game-Server	29
5.3	Lösungsansatz 2: Emulation im Browser	30
5.4	Zusammenfassung	31
6	Methodik	33
7	Konzeption	35
7.1	Entwurf der Benutzeroberfläche	36
7.2	Software-Architektur	36
8	Implementierung	37
8.1	Entwicklungsumgebung	38
8.2	Verwendete Software-Komponenten	38
8.3	Ausgewählte Implementierungsdetails	38
8.4	Vollständigkeit	39
9	Evaluation	41
9.1	Aufbau der Messumgebung	43
9.2	Ergebnisse und Beobachtungen	43
9.3	Diskussion und Bewertung	43
10	Fazit	45
10.1	Zusammenfassung	46
10.2	Bewertung	46
10.3	Ausblick	46

I Verzeichnisse	49
Abbildungsverzeichnis	51
Literatur	53
Glossar	57
Akronyme	59
II Anhang	61
Entwürfe	63

1

Einleitung

“Begin at the beginning,” the King said, gravely, “and go on till you come to an end; then stop.”

— Lewis Carroll, *Alice in Wonderland*

Nach der Krise der Videospiel-Industrie Mitte der achtziger Jahre hatte Nintendo mit der Veröffentlichung des **Nintendo Entertainment System (NES)** eine Konsole für Videospiele auf den Markt gebracht, die sich millionenfach verkaufte. Dabei führte das Unternehmen weltbekannte Marken wie Mario, Donkey Kong und Zelda ein, mit denen das Unternehmen bis zum heutigen Tag erfolgreich ist. In den 90er Jahren wurde der Nachfolger des **NES** auf den Markt gebracht, das **Super Nintendo Entertainment System (SNES)**. Beide Systeme verkauften sich zusammen weltweit über 100 Millionen mal und sind damit die erfolgreichsten Systeme ihrer Generation. Insgesamt wurden für beide Systeme ca. 1495 Spiele entwickelt und lizenziert; darunter Singleplayer-

Klassiker wie Super Metroid oder »Mega Man«. Gespielt werden konnte aber nicht nur allein: Die Konsolen wurden von vornherein für mehrere Spieler ausgelegt und verfügen über mehrere Controller-Anschlüsse zur Eingabe zur Spielsteuerung. Dies führte dazu, dass viele Spiele sowohl einen Singleplayer- als auch einen Multiplayer-Modus beinhalteten, der es ermöglichte, mit mehreren Spielern an der Konsole zu spielen.

Mittlerweile haben sich mindestens zwei weitere erfolgreiche Videospiel-Konsolen im Markt etabliert: Die Xbox One von Microsoft und die PlayStation 4 von Sony. Wie auch schon das [NES](#) unterstützen beide Systeme mehrere Spieler über Controller, die mit der Konsole gekoppelt sind. Die technologische Entwicklung seit den 80er und 90er Jahren ist gut am Internet abzulesen: Üblich waren Internetanschlüsse mit einer Bandbreite von weit unter einem Megabit. Heutzutage sind breitbandige Internetanschlüsse großflächig verfügbar. Dieser technologische Fortschritt ermöglicht es, größere Datenmengen übers Internet zu übertragen – z. B. die Daten eines Multiplayer-Spiels. Man spielt nicht mehr exklusiv im eigenen Wohnzimmer oder auf einer LAN-Party gemeinsam miteinander, sondern kann dies über das Internet und über Landesgrenzen hinweg tun. Dazu haben Microsoft und Sony jeweils kostenpflichtige Online-Plattformen geschaffen, über die Spieler miteinander über verschiedene Kanäle interagieren – und, viel wichtiger – miteinander spielen können. Die Plattformen bieten einen Treffpunkt der Kommunikation und Vermittlung und stellen für die Spiele-Entwickler Dienste wie Spiel-Lobbys und Matchmaking bereit.

Formulierung
und Quelle

Eine bemerkenswerte Eigenschaft beider Plattformen ist deren einfache Handhabung: Sie machen es den Spielern sehr einfach, gemeinsam über das Internet miteinander zu spielen. Die Plattformnutzer müssen über keine speziellen Kenntnisse verfügen, müssen keine zusätzliche Software installieren oder etwa darin geübt sein, einen dedizierten Game-Server aufsetzen und zu administrieren. Mitglieder der Plattform können sich auf das Spielen konzentrieren. Aktuelle Spiele-Titel können ohne große Hürden über die beschriebenen Plattformen gespielt werden. Eine Plattform für [SNES](#)-Spiele, die einen vergleichbaren Komfort bietet, existierte bislang nicht.

1.1 Motivation

Möchte man am PC ein Konsolenspiel, z. B. Super Mario, spielen, benötigt man neben dem eigentlichen Spiel auch ein Programm, mit dem das Spiel ausgeführt werden kann: einen Emulator. Einen geeigneten Emulator auszuwählen ist nicht einfach: Für das **SNES** allein existieren mehr als ein Dutzend Emulatoren für verschiedene Betriebssysteme (vgl. Wikipedia, *List of video game emulators*), die sich in der Genauigkeit der Emulation, der Kompatibilität und weiteren Aspekten unterschieden und von denen partiell zusätzliche Ableger (Forks) existieren. Ein Teil dieser Emulatoren verfügt über einen Multiplayer-Modus. Mit dem Problem, dass die verschiedenen Emulatoren untereinander meist inkompatibel sind. Auch bei komplexeren Emulationssystemen wie RetroArch, das viele Emulatoren enthält und um eine eigens geschaffene Netplay-Funktion erweitert, ist die Kompatibilität nicht garantiert (vgl. Richards, *Netplay core testing*). Um erfolgreich ein **SNES**-Spiel im Multiplayer-Modus gespielt werden, müssen alle Spieler im Idealfall den gleichen Emulator in der selben Version verwenden. Sofern die verwendeten Emulatoren untereinander kompatibel sind, existieren weitere Hürden, die für einen Anwender ohne technischen Hintergrund nicht ohne Weiteres zu lösen sind. Ein möglicher Ablauf zum Erstellen eines Spiels sieht beispielsweise so aus:

1. Einer der Spieler wird als Spielleiter (**Host**) bestimmt und startet den Emulator im Netzwerk-Modus.
2. Der Host muss dabei sicherstellen, dass der spezifische Port auf dem Emulator-Prozess läuft von allen Mitspielern über das Netzwerk erreichbar ist. Falls der Host-Spieler sich in einem Netzwerk befindet, dessen Routing auf **Network Address Translation, RFC 1631 (NAT)** basiert, muss eine Port-Weiterleitung für den Emulator-Prozess eingerichtet werden, der auf dem PC des Host-Spielers läuft. Dazu muss die verwendete Portnummer des Emulators bekannt und eine Port-Weiterleitung im Administrationsbereich des Routers konfiguriert sein. Für das beschriebene Szenario ist die korrekte **NAT**-Konfiguration Voraussetzung für das Spielen im Multiplayer-Modus.
3. Wenn der ausgewählte Emulator über keine Funktion verfügt, um Netzwerkspiele aufzulisten, benötigen alle Mitspieler die IP-Adresse des Host-Spielers, um eine Verbindung herzustellen.
4. Der Host-Spieler muss die eigene öffentliche IP-Adresse ermitteln und seinen

Mitspielern mitteilen.

5. Die Mitspieler verbinden sich mit dem Host-Spieler über die Eingabe der IP-Adresse des Hosts im entsprechenden Dialog der Benutzeroberfläche des verwendeten Emulators.

1.2 Zielsetzung

Im Zuge der immer besseren Verfügbarkeit von breitbandigen Internetanschlüssen entstehen neue Chancen für die Entwicklung neuer und interessanter Lösungen. Mit den entstandenen Chancen ergeben sich aber auch neue Herausforderungen, für deren Lösung neue Technologien geschaffen werden müssen. Ein technologischer Bereich, der sich besonders schnell weiterentwickelt, ist das das [World Wide Web \(Www\)](#), das spätestens seit Beginn der Web 2.0-Ära einen riesen Aufschwung erfährt und sich die Computer-Nutzung immer mehr in den Internet-Browser und das [Www](#) verlagert.

Ablesen lässt sich die Weiterentwicklung der Web-Technologien zum Beispiel durch die große Anzahl der neuen Standards- und Protokolle. Der Internet-Browser verfügt nahezu über den Funktionsumfang eines Desktop-Systems und kann ein breites Spektrum von Anwendungsfällen abdecken. Das Abspielen von Audio- und Videodaten, die direkte Kommunikation zwischen Browsern und das Ausführen von komplexen 3D-Anwendungen ist mittlerweile kein Problem mehr.

Diese Arbeit untersucht die Frage, ob die aktuell verfügbaren Web-Standards die Anforderungen für die Entwicklung einer Multiplayer-Plattform für [SNES](#)-Spiele erfüllen und welche Technologien in welcher Weise kombiniert werden müssen, um ein funktionales Basis-System zu schaffen. Die zentralen Aspekte sind die Auswahl, die Kombination und die Integration der verschiedenen Technologien zu einem Gesamt-System bestehend aus verschiedenen Komponenten, die alle im Zuge dieser Arbeit entwickelt werden.

Der Fokus liegt dabei auf der Entwicklung einer funktionalen Web-Plattform mit den wichtigsten Basisfunktionen, die aufgrund einer sauberen Architektur leicht um neue Funktionen zu erweitern ist.

1.3 Vorgehensweise

1. **Grundlagen** Einarbeitung und Erläuterung der technischen Grundlagen, die für die Lösung der Zielsetzung notwendig sein können. Dies umfasst verschiedene Web-Technologien auf der einen Seite und Emulationssysteme auf der anderen Seite.
2. **Spezifikation** Genaue Festlegung des Funktionsumfangs des zu erstellenden Systems.
3. **Herausforderungen** Ableitung der zu lösenden Teilprobleme, die sich aus den definierten Zielen ergeben.
4. **Lösungsansätze** Untersuchen, ob und welche Teillösungen für die im vorherigen Abschnitt beschriebenen Probleme existieren. Vorstellen von Lösungsansätzen, die ggf. auf vorhandenen Teillösungen aufbauen oder diese erweitern. Gewählten Lösungsansatz vorstellen und mögliche Alternativen aufzeigen.
5. **Methodik** Beschreibung der geplanten Umsetzung.
6. **Konzeption** Entwurf der Benutzeroberfläche, Definition der System-Komponenten und der Software-Architektur.
7. **Implementierung** Technische Beschreibung der entwickelten Lösung mit Fokus auf ausgewählte Aspekte der Implementierung.
8. **Evaluation** Überprüfung und Bewertung der Lösung im Hinblick auf das Erreichen der definierten Ziele.
9. **Fazit** Zusammenfassung und Bewertung der Arbeit sowie Vorstellen von Erweiterungsmöglichkeiten.

2

Grundlagen

Inhalt dieses Kapitels ist eine Einführung in die technischen Grundlagen dieser Arbeit. Es werden die wichtigsten Aspekte und Technologien erläutert und für das Lösungsumfeld essentielle Begriffe erklärt. Die Erläuterungen schaffen ein Grundverständnis für die in den nachfolgenden Kapiteln behandelten Aspekte und sind für eine Beurteilung der entwickelten Lösung notwendig. Schwerpunkte sind für die Zielsetzung potentiell geeignete Web-Technologien, Emulatoren & ROMs und Grundlagenthemen zur Kommunikation in Computer-Netzwerken.

2.1 Emulation von Konsolenspielen

Wie eingangs erwähnt, sind zum Spielen von SNES-Spielen am Computer zwei Dinge notwendig: Ein Emulationsprogramm — auch Emulator genannt — und das jeweilige Konsolenspiel, das ausgeführt werden soll. Das Spiel liegt dabei in Form einer Datei vor. Sie beinhaltet die exakt gleichen Daten, die sich für gewöhnlich auf den Modulen befinden, die zum Spielen in die Konsolen gesteckt werden müssen. Zum Kopieren der Daten existieren spezielle Geräte, mit denen eine solche Datei erzeugt werden kann. Dabei wird ein genaues Abbild des Moduls erzeugt, inklusive der Sektoren und der jeweiligen Dateisystem-Struktur. Die Dateien werden oft als ROMs bezeichnet, da die Daten des Spiels innerhalb des Moduls in einem ROM-Chip gespeichert sind. Um ein Spiel auszuführen zu können, muss die Spiel-Datei in den Emulator geladen werden.

Der Emulator ist ein Software-Programm, das es ermöglicht Konsolentitel auf einem Computer auszuführen. Dies funktioniert, indem die Hardware der jeweiligen Konsole nachgeahmt wird. Die verschiedenen Komponenten der Konsole (CPU, Soundchip, etc.) sind dabei so in Software implementiert, dass sie sich möglichst so verhalten, wie die nachzuahmende Hardware. Erst eine möglichst exakte Emulation der ursprünglichen Laufzeitumgebung ermöglicht die Ausführung von unveränderten Spielen in Form von ROM-Dateien. Die Spiele, bzw. das Abbild der Software, ist eine exakte Kopie des Originals.

- Roms
- Emulatoren

2.2 Der Multiplayer-Modus

Im Netzwerk! Im Gegensatz zum lokalen Multiplayer.

Welche Eigenschaften/Merkmale hat der lokale Multiplayer, die auch im Netzwerk-Multiplayer gelten sollen

Das erste entwickelte Computerspiel war ein Multiplayer-Spiel und benötigte damit mindestens zwei Spieler, die gemeinsam an einem Gerät spielen. Die Verarbeitung der Benutzereingaben erfolgt zentral an dem einzigen Gerät, so wie auch die Darstellung der grafischen Oberfläche.

2.3 Grundbegriffe der Netzwerk-Kommunikation

Evtl. ganz an den Anfang schieben, damit man im Multiplayer-Abschnitt darauf verweisen kann.

2.3.1 Latenz

- Minimalste Latenz essentiell beim Spielen
- Alles über N ms ist nicht gut genug. Um angepeilten Maximalwert zu erreichen, System entsprechend konstruieren und Technologien wählen, die eine hinreichende Lösung erst möglich machen.
- Wo von hängt Latenz ab? Was kann man da überhaupt machen?
- RTT, TCP vs. UDP?

2.3.2 RTT

2.3.3 Realtime

2.4 Das Realtime-Web

Vergleich zur alten Herangehensweise. Frontend viel Logik, Backends eigentlich nur noch API.

2.4.1 Websockets

2.4.2 HTTP/2

2.4.3 SSE

2.4.4 WebRTC

Hier nur beschreiben. Hier kann ich letztlich auch nur raten, dass die gewählten Technologien sinnvoll sein werden. Ob die getroffene Auswahl gut war (wurden Ziele erreicht?), zeichnet sich ggf. schon während der Implementierung ab. Ergebnisse und Erkenntnisse, die die Bildung eines ersten Urteils ermöglichen, werden erst in der Evaluationsphase geschaffen.

NICHT: Vergleichen/Abwägen, Entscheiden, Begründen

3

Spezifikation

Dieser Abschnitt definiert die zu erfüllenden System-Anforderungen, deren Erreichen im späteren Verlauf dieser Arbeit während der Evaluation überprüft werden. Zunächst wird definiert, über welche Grundfunktionen das System verfügen soll. Weiter gelistet werden die Leistungs- und Qualitätsanforderungen sowie die Randbedingungen, innerhalb derer die definierten Anforderungen gelten. Abgeschlossen wird das Kapitel durch eine Auflistung der Kriterien, die explizit *nicht* Gegenstand dieser Arbeit sind.

Kurze Einordnung
Völlig neu? Alt?
Einfache Lösung? Total schwer?
Forschung? Anwendung?

Die [SNES](#)-Konsole unterstützt grundsätzlich nur *lokalen Multiplayer* mit bis zu zwei — mit speziellem Adapter vier — Spielern. Die Eingabe der Befehle erfolgt über mitgelieferte Eingabegeräte ([Controller](#)), die jeweils über ein Kabel mit der Konsole verbunden sind. Über einen Multiplayer-Modus zum Spielen über das Internet, wie er von heutigen Konsolen unterstützt wird, verfügt die Konsole nicht.

Ursprünglich für diesen Anwendungszweck nicht vorgesehen, wird im Zuge der Arbeit ein System entwickelt, das das Spielen von [SNES](#)-Spielen über das Internet ermöglicht. Die einzelnen Spieler sind dabei nicht über eine Konsole, sondern über einen Web-Browser miteinander verbunden (siehe Abbildung 3.1). Die Verbindung wird dabei über eine Web-Plattform, nachfolgend auch Web-Applikation (kurz Web-App), initiiert und verwaltet.



Abbildung 3.1: Zum Spielen ist ausschließlich ein Web-Browser notwendig.

Die große Herausforderung dieser Arbeit besteht darin, das Spielerlebnis des [SNES](#) um einen neuen Multiplayer-Modus zu erweitern, der das gemeinsame Spielen über das Netzwerk überhaupt erst ermöglicht. Bereits bestehende Emulations-Programme lösen das Problem in Ansätzen, in der Praxis ist deren Verwendung aber unnötig komplex und benötigt neben dem eigentlichen Spiel weitere Software und eine spezielle Netzwerk-Konfiguration (siehe Abschnitt).

Um die Grundfunktionen des zu erstellenden Systems zu nutzen, wird außer einem aktuellen Web-Browser *keine* spezielle Software benötigt. Das Spiel wird direkt im Browser ausgeführt, wobei die Steuer-Befehle über die jeweils angeschlossene Computer-Tastatur entgegengenommen und verarbeitet werden.

Alternativ haben Benutzer die Möglichkeit die Web-Applikation zusätzlich über ihr Smartphone zu öffnen. Die Web-Applikation erkennt das entsprechende Gerät und bietet für Smartphones einen anderen Funktionsumfang an, als in der Desktop-Version.

Nach dem Koppeln der Web-App und dem Smartphone erfolgt die Eingabe-Verarbeitung nicht mehr über die Computer-Tastatur, sondern über den auf dem Smartphone dargestellten virtuellen **Controller**. Das Prinzip ist in Abbildung 3.5 dargestellt.

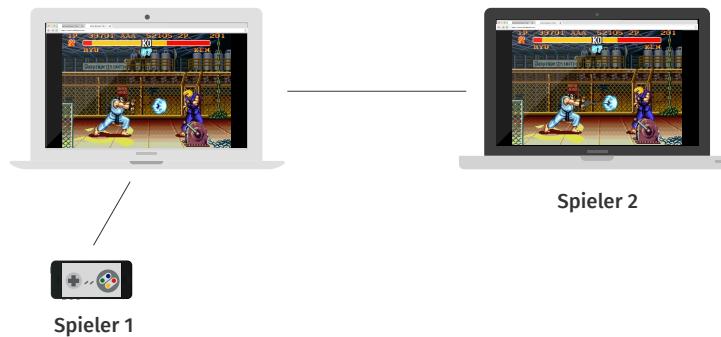


Abbildung 3.2: Bei Verwendung der Controller-App erfolgt die Eingabe der Steuerbefehle nicht mehr über den Web-Browser, sondern über das mit dem Web-Browser gekoppelte Smartphone.

Neben ihrer Funktion als "besserer Controller" schafft die Smartphone-Variante außerdem die Möglichkeit, dem laufenden Spiel weitere (lokale) Spieler hinzuzufügen. Das bedeutet, dass pro Web-Browser mehrere Spieler spielen können, die sich einen gemeinsamen Bildschirm teilen. Das Hinzufügen der lokalen Spieler funktioniert dabei so wie die Kopplung des Smartphones von Spieler 1 und muss vom Spielleiter initiiert werden (siehe Abbildung 3.3). Durch diese Flexibilität sind verschiedene Spiel-Konfigurationen möglich.



Abbildung 3.3: Hohe Flexibilität: Spiel-Session mit einem entfernten (rechts) und zwei lokalen Spielern, die sich einen Bildschirm teilen (links).

3.1 Funktionale Anforderungen

Dieser Abschnitt beschreibt die Spezifikation des Soll-Zustands des zu erstellenden Systems und seiner Komponenten. Der Soll-Zustand wird während der Evaluation überprüft, in dem ein Vergleich zwischen Soll- und Ist-Zustand vorgenommen wird.

Das zu entwickelnde System besteht aus den Komponenten **Web-App** und **Controller-App**, deren Anforderungen nachfolgend aufgelistet sind.

3.1.1 Web-Applikation

Die Web-Applikation besteht aus vier verschiedenen Ansichten, die untereinander verknüpft sind und zwischen denen der Benutzer während der Nutzung navigiert:

- **Startseite**,
- **Authentifikation**,
- **Lobby**,
- und **Match**.

Abbildung 3.4 gibt einen Überblick über die verschiedenen Ansichten und Navigationspfade der Web-Applikation. Die in den jeweiligen Ansichten enthaltenen Funktionen werden nachfolgend erläutert und bilden zusammen die funktionalen Anforderungen, die an die Web-Applikation gestellt werden.

Ansicht 1: Startseite

Die Startseite ist keine vom Benutzer sichtbare Ansicht. Die Funktion der Startseite ist lediglich die Weiterleitung des Benutzers auf eine nachfolgende Ansicht. Ist der Benutzer dem System bekannt, erfolgt eine Weiterleitung auf die Lobby-Ansicht. Ist der Benutzer unbekannt, ist eine Authentifikation notwendig und der Benutzer wird zur Authentifikation-Ansicht weitergeleitet.

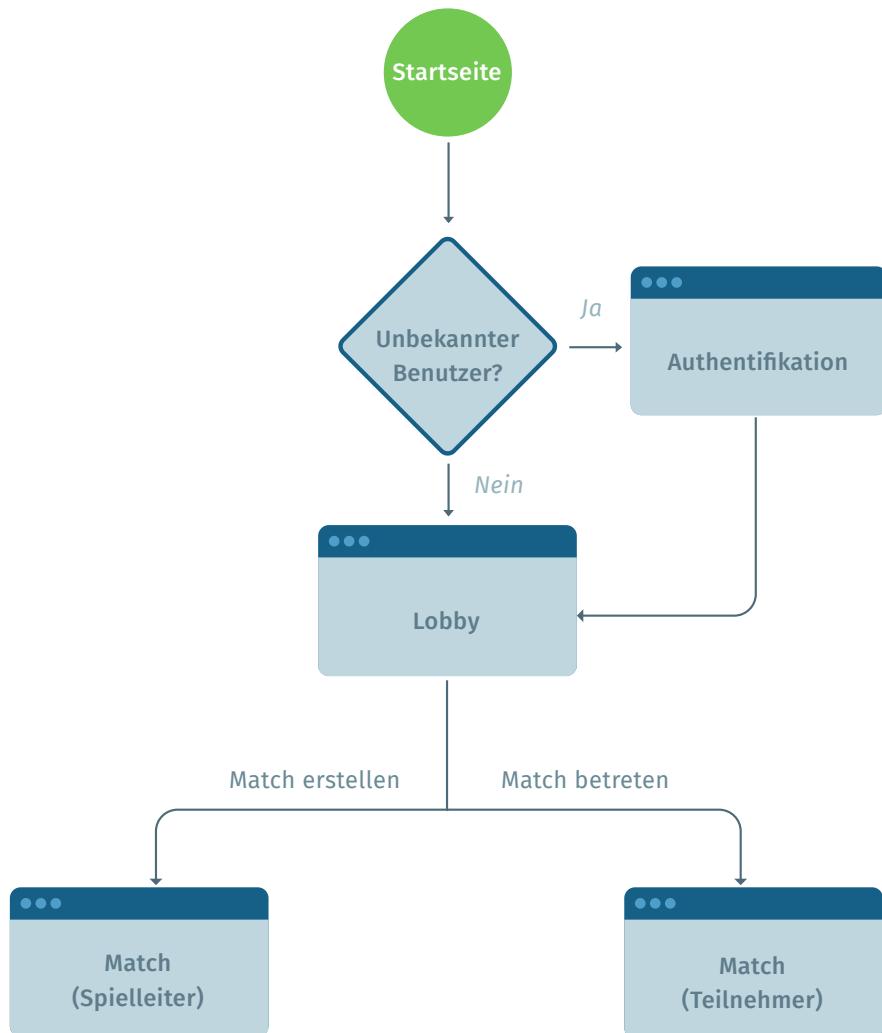


Abbildung 3.4: Ansichten und Navigationspfade der Web-Applikation.

Ansicht 2: Authentifikation

Voraussetzung für die Nutzung der Web-Applikation ist die Authentifikation des Benutzers. Dabei wird der Benutzer aufgefordert einen Benutzernamen zu vergeben. Die Eingabe erfolgt über ein herkömmliches Web-Formular. Wird kein Benutzername vergeben, erhält der Benutzer einen vom System zufällig generierten Benutzernamen. Benutzernamen müssen nicht eindeutig sein – die Identifikation erfolgt über eine eindeutige Benutzer-ID, die nicht angezeigt wird. Nach der Verarbeitung des Benutzernamens wird dieser, zusammen mit weiteren Informationen, im Browser des Nutzers gespeichert und die Web-Applikation wechselt zur Lobby-Ansicht.

Ansicht 3: Lobby

Nach der Authentifikation des Benutzers und dessen Weiterleitung auf die Lobby-Ansicht werden dem Benutzer folgende Funktionen zur Verfügung gestellt:

Auflisten der aktuellen Matches Nach der Authentifikation wird der Benutzer auf die Lobby-Ansicht weitergeleitet. Die Lobby ist eine Liste aller Matches, die von anderen Benutzern erstellt worden sind und denen der Benutzer beitreten kann, um an Matches teilzunehmen. Aufgelistet werden nur jene Matches, deren maximale Spielerzahl noch nicht erreicht ist. Die Liste zeigt den aktuellen Stand der Matches in Echtzeit an und wird automatisch aktualisiert, ohne dass das Browser-Fenster neu geladen werden muss.

Erstellen (“Hosten”) eines neuen Matches Der Benutzer muss die Möglichkeit haben, ein neues Match zu erstellen, um entweder allein oder gemeinsam mit anderen Personen zu spielen. Voraussetzung zum Erstellen eines neuen Matches ist die Bereitstellung des Spiels in Form einer **ROM**-Datei, die vom Benutzer im Verlauf der Spielerstellung über ein Web-Formular hochgeladen werden muss. Neben der **ROM**-Datei muss ein Titel für das Match vergeben und die maximal gewünschte Spielerzahl festgelegt werden. Unterstützt werden sollen pro Match ein bis maximal vier Spieler. Die ursprünglich für das Spiel vorgesehene maximale Spielerzahl des hochgeladenen Spiels wird nicht ermittelt. Es liegt im Ermessen des Benutzers eine für das jeweilige Spiel sinnvolle¹ maximale Spielerzahl festzulegen. Solange das Match existiert und der Benutzer das Spiel durch Schließen des Browser-Fensters nicht verlässt, hat er die Rolle des Spielleiters. Als solches stehen ihm für das erstellte Match mehr Funktionen zur Verfügung als den Benutzern, die dem Spiel als Mitspieler beitreten. Die Beschreibung der Funktionen befindet sich in Abschnitt [Ansicht 3: Match](#).

Match beitreten Der Benutzer soll einem bestehenden Match beitreten können, um daran teilzunehmen. Vor dem Beitreten werden keine Informationen abgefragt, das jeweilige Match muss nur ausgewählt werden. Grundsätzlich können alle Benutzer jedem gelisteten Match beitreten. Liegt die vom Spielleiter konfigurierte maximale Spielerzahl über der vom Spiel unterstützten, betritt der Benutzer das Match im Zuschauer-Modus und kann nicht ins Spielgeschehen eingreifen. Vor dem Betreten wird dem Benutzer nicht angezeigt, in welchem Modus (als

¹Zum Beispiel zwei für “Street Fighter 2” und vier für “Super Bomberman”.

Mitspieler oder als Zuschauer) das Match betreten wird.

Ansicht 3: Match

Die Match-Ansicht setzt sich aus mehreren Bereichen zusammen, über die unterschiedliche Funktionen zugänglich gemacht werden.

Die **Leinwand** ist der prominente Bereich der Ansicht, in dem das laufende Spiel dargestellt wird.

Spiel anhalten, pausieren und neustarten Der Spielleiter kann das Spiel über Schaltflächen starten, pausieren und neustarten.

Spielen im Vollbildmodus Jeder Mitspieler hat hier die Option, die Leinwand über eine Schaltfläche in den Vollbildmodus umzuschalten. Der Vollbildmodus kann über die ESC-Taste verlassen werden.

Die **Spielerliste** listet alle im Match befindlichen Benutzer auf. Die Liste wird stets aktualisiert und zeigt jedem Benutzer in Echtzeit die aktuellen Mitspieler an.

Entfernen eines Spielers aus einem Match Der Spielleiter kann beliebige Spieler über eine Schaltfläche aus dem Match entfernen (“kicken”). Nach dem Entfernen wird der Benutzer auf die Lobby-Ansicht weitergeleitet. Eine Sperrzeit ist nicht vorgesehen. Der Spieler kann dem Match theoretisch erneut beitreten.

Den letzten Bereich der Match-Ansicht bildet der **Chat**, bestehend aus dem Chat-Verlauf und einem Formular zum Senden einer neuen Chat-Nachricht. Der Chat-Verlauf wird nichtpersistiert: Jedem Benutzer werden nur diejenigen Nachrichten angezeigt, die nach dem Zeitpunkt seines Beitrags zum Match verfasst worden sind.

Match-eigener Chat Über den Chat-Bereich hat jeder Benutzer die Möglichkeit, Nachrichten mit den Mitspielern aus dem Match auszutauschen. Chat-Nachrichten werden über ein Formularfeld eingegeben und über eine Schaltfläche abgesendet. Der Chat-Verlauf aktualisiert sich in Echtzeit. Alle im Match befindlichen

3 Spezifikation

Benutzer (inkl. Zuschauer) können am Chat teilnehmen und bekommen alle Nachrichten in Echtzeit angezeigt.

Wenn ein Benutzer ein Match durch Schließen des Browserfensters oder durch die Navigation auf eine andere Seite verlässt, wird die Spielerliste bei allen Mitspielern live aktualisiert. Verlässt der Spielleiter das Match, wird das Match beendet und alle Mitspieler werden auf die Lobby-Ansicht weitergeleitet.

Voraussetzung für ein netzwerkbasierteres Multiplayer-Spiel ist eine aktive Verbindung zwischen allen Mitspielern eines Spiels. Wie diese Verbindung zustande kommt, ist von System zu System unterschiedlich. Bei den meisten Emulatoren ist es notwendig, einen Spieler als Spielleiter zu bestimmen. Das Emulator-Programm dieses Spielers wird dabei im Host-Modus gestartet und bietet einen Netzwerkdienst an, der über das Internet erreichbar sein muss und deren Adresse den Mitspielern mitgeteilt werden muss. Dieses Verfahren hat mehrere Nachteile, die bereits in Kapitel erläutert wurden. Eine bessere Lösung, die sich mittlerweile auch durchgesetzt hat, ist das Konzept einer Spiel-Lobby, in dem alle laufenden Spiele aufgelistet sind. Die Lobbys verfügen meist über Such- und Filterfunktionen zum Finden des gesuchten Spiels. Die Lobby ist eine Lösung für das *Matchmaking* und erleichtert das, in dem es die Komplexität des Vorgangs, die im erstgenannten Ansatz vorhanden ist (IP-Adressen, Netzwerk-Konfiguration, Firewalls, etc.), vor den Spielern verbirgt.

Passt wahrscheinlich besser woanders hin

3.1.2 Controller-Applikation

Die Controller-Applikation ist keine native Smartphone-App, sondern ein Modul der Web-Applikation. Sie wird angezeigt, sobald die Web-Applikation mit einem Smartphone über einen Web-Browser geöffnet wird. Da der Funktionsumfang sich komplett von der Web-Applikation unterscheidet, ist eine Unterteilung beider Applikationen sinnvoll.

Die Controller-App soll die Eingabe der Steuerbefehle beim Spielen erleichtern, in dem die verschiedenen Tasten des **SNES**-Controllers auf einem Smartphone dargestellt werden. Die Tasten werden über ein Tippen auf die Smartphone-Oberfläche betätigt und zur Web-Applikation übertragen, wo deren eigentliche Verarbeitung stattfindet.

Für ihren Einsatz muss Controller-App mit der Web-App gekoppelt werden. Initiiert wird die Kopplung über die Oberfläche der Web-Applikation. Dazu wird auf der Web-Applikation ein QR-Code angezeigt, der über das Smartphone und die Controller-App gescannt werden muss. Nach dem Scannen ist der Kopplungsvorgang abgeschlossen. Jeder Spieler kann sein eigenes Smartphone koppeln und damit seine Spielfigur über das Smartphone steuern.

Weiter soll es jedem Spieler ermöglicht werden, über eine Schaltfläche, die auf der Web-Applikation dargestellt wird, einen weiteren lokalen Mitspieler hinzuzufügen, sofern das Spiel weitere Spieler unterstützt — die maximale Spielerzahl wird vom Spielleiter während der Spielerstellung festgelegt. Das Hinzufügen des lokalen Mitspielers ist identisch zu dem beschriebenen Kopplungsvorgang: Anzeige und Scan in der Web-Applikation angezeigten QR-Codes. Nach der Kopplung ist der lokale Mitspieler Teil des Spiels.

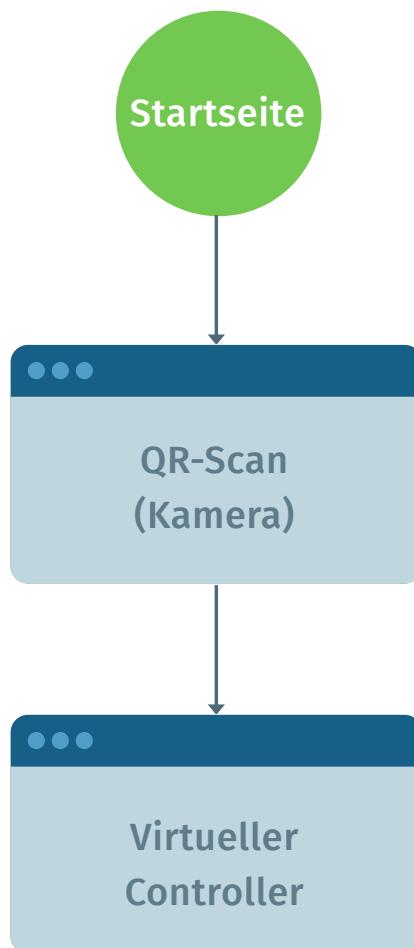


Abbildung 3.5: Controller-App

Die Controller-App besteht aus zwei Ansichten, deren Funktionen gemeinsam den gesamten Funktionsumfang der App ergeben.

3.1.2.1 Ansicht 1: Startseite

Die Startseite wird angezeigt, sobald der Benutzer die Web-Applikation auf einem Smartphone öffnet. Die Startseite beinhaltet eine Schaltfläche zum Scannen eines QR-Codes. Wird diese angetippt, wird Ansicht 1 angezeigt.

3.1.2.2 Ansicht 1: QR-Scan

Die Scan-Ansicht öffnet die Kamera-Ansicht des Smartphones und bietet die Möglichkeit zum Scannen des von der Web-App dargestellten QR-Codes. Dazu muss der QR-Code wie üblich mit der Kamera erfasst werden. Sobald der QR-Code ausgelesen wurde, erfolgt die Kopplung mit der Web-Applikation und die Weiterleitung zur nächsten Ansicht erfolgt.

3.1.2.3 Ansicht 2: Virtueller Controller

Diese Ansicht wird dargestellt, sobald die Controller-App mit der Web-Applikation gekoppelt ist. Sie dient nachfolgend als Eingabegerät, über das das Spiel gesteuert wird. Hier werden die Buttons des **SNES**-Controllers über entsprechende Schaltflächen nachgeahmt. Jeder Fingertipp auf eine Schaltfläche der Controller-App wird an die Web-Applikation weitergeleitet.

3.2 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen beschreiben die Qualitätbedingungen des zu entwickelnden Systems. Sie werden ebenfalls während der Evaluation überprüft.

3.2.1 Leistung- & Qualitätsanforderungen

Die Emulation der **SNES**-Spiele im Browser ist das Herz des Systems und die wichtigste Funktion. Die Spiele sollen flüssig laufen, die Verzögerung zwischen der Ausgabe von Video und Audio soll minimal sein. Der erste Aspekt kann über die Metrik Frames gemessen werden - der Wert beschreibt die Anzahl Bilder pro Sekunde. Der Wert sollte nicht unter 60 liegen. Die Audiolatenz soll so gering wie möglich sein, eine Verzögerung zwischen Bild und Ton hängt von der Implementation und Konfiguration des Emulators sowie des Computers ab, auf dem die Emulation läuft. Ein Wert von unter 100 ms wird angestrebt.

<https://gamedev.stackexchange.com/questions/74973/maximum-audio-delay-before-the-player-notices>

Ein weiterer wichtiger Aspekt ist die Netzwerk-Latency zwischen den einzelnen Spielern und den von Ihnen verwendeten Web-Browsern. Für ein gutes Spiel-Erlebnis muss die Latency so gering wie möglich sein. Die System-Architektur muss entsprechend darauf hin optimiert sein. Die Höhe der Netzwerk-Latency ist aber nicht nur abhängig von der gewählten System-Architektur und der konkreten Implementation, sondern hängt in erster Linie von physikalischen Faktoren ab, auf die Software-Entwickler keinen Einfluss haben: Je weiter zwei Spieler geographisch voneinander entfernt sind, desto höher ist die Latency zwischen ihnen. Ein Spiel zwischen zwei Berlinern wird eine geringe Latency aufweisen als ein Spiel zwischen einem Berliner und einem Spieler aus Paris. Einen absoluten Zielwert für die Netzwerk-Latency zu nennen ist also nicht sinnvoll. Um trotz dessen eine Aussage zur Qualität bzgl. diesen Aspekts zu machen, wird für die Evaluation vorausgesetzt, dass sich alle Spieler eines Spiels im selben Subnetz des selben Netzwerks befinden – also z. B. im selben WLAN eines Routers. Die Netzwerk-Latency ist unter diesen Voraussetzungen für alle Teilnehmer vergleichbar gering. Diese Einschränkung ermöglicht es, die Netzwerk-Latency auszuklammern und den Fokus auf die Untersuchung zu legen, wie viel Latency vom System zusätzlich erzeugt wird.

3.2.2 Anforderungen an die Umgebung

Alle definierten Anforderungen gelten nur im Kontext der hier beschriebenen Umgebung — die Lauffähigkeit des entwickelten Systems ist ebenfalls nur dann gegeben.

3.2.2.1 Nutzungsbedingungen

Grundsätzlich unterstützt das System alle modernen Browser, die folgende Standards vollständig und fehlerfrei unterstützen:

1. JavaScript
2. [Canvas](#)-Element
3. WebGL
4. WebRTC (inkl. RTCPeerConnection, MediaCapture)
5. WebAssembly
6. WebAudio
7. WebSockets

Glossar-
Links

Installierte Browser-Erweiterungen², die die o.g. Technologien blockieren, deaktivieren oder die Funktionsweise einschränken, müssen deaktiviert sein. Theoretisch erfüllt und deswegen angestrebt ist eine Unterstützung der Desktop-Varianten der Web-Browser Google Chrome und Mozilla Firefox — unter den genannten Voraussetzungen. Die Browser verfügen meist über unterschiedliche Implementationen der genannten Standards, wobei es große Unterschiede bzgl. Funktionsumfang und Korrektheit geben kann. Eine qualitativ gleichwertige Kompatibilität für mehrere Browser zu schaffen ist kein triviales Problem, zu dessen Lösung innerhalb dieser Arbeit keine Zeit zur Verfügung steht. Im Zuge der Umsetzung wird ein Prototyp implementiert, der es möglich macht, sich einen ersten Eindruck zur Kompatibilität zu verschaffen. Das System wird anschließend auf diejenigen Web-Browser hin optimiert, die bei einem manuellen Vergleich die höchste Kompatibilität haben und die Software standardkonform ausführen.

warum er-
füllen die
beiden das?
Kompatibili-
tätsmatrizen,
iswebrtcrea-
dyet

²Zum Beispiel „uBlock Origin Extra“, das diverse WebRTC-Funktionen deaktiviert.

Browser-Liste, die die Anforderungen im Moment erfüllen?

Um die Controller-App nutzen zu können, muss das Smartphone über eine Kamera verfügen.

3.2.2.2 Betriebsbedingungen

Das entwickelte System ist eine komplette Web-Anwendung („Full Stack“) und benötigt neben verschiedenen Software-Bibliotheken keine weiteren Module, die außerhalb des Kontextes dieser Arbeit entstanden sind. Voraussetzung für den Betrieb ist ein Linux-Betriebssystem mit Unterstützung für Docker. Alle benötigten Software-Pakete werden automatisch innerhalb der Docker-Umgebung installiert und konfiguriert.

Meine eigenen, evtl. vorher entwickelten, Third-Party natürlich.

Das System ist nicht auf maximale Skalierbarkeit hin optimiert, weil dies außerhalb des Kontextes dieser Arbeit liegt. Anfragen kleinerer Benutzergruppen von ca. 10-20 gleichzeitigen Nutzern, die gegebenenfalls während einer Testphase verarbeitet werden müssen, sollen sich nicht negativ auf die beschriebene Qualität auswirken.

3.2.3 Randbedingungen & Abgrenzungskriterien

Dieser Abschnitt beschreibt Funktionen, die nicht Teil der Implementation des Systems sind bzw. Kriterien, die – meist aufgrund der begrenzten Zeitfensters – nicht an das System gestellt werden können.

Benutzverwaltung Für das System wird auf eine herkömmliche Benutzerverwaltung mit Registrierung & Login verzichtet. Der Nutzer wird einzig über eine eindeutig generierte Identifikationsnummer authentifiziert, die im Browser hinterlegt wird. Geht diese verloren, erhält der Spieler beim nächsten Besuch eine neue Nummer.

Netzwerk-Latenz Das System implementiert kein System zur Kompensation der Latenz (dies beinhaltet jegliche Latenz: Netzwerk, Audio/Video, weitere).

Accessibility Eine Optimierung der Accessibility des Systems findet nicht statt.

4

Probleme & Lösungsansätze

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4.1 Web-Applikation in Echtzeit

- Stateless Request/Response-Modell macht aufgrund der Anforderungen keinen Sinn
- Man könnte selektiv z.B. AJAX / Long Polling etc. machen. Macht aber wenig Sinn, weil man, wie man später sieht, WebRTC braucht und man dafür einen Signaling Channel bauen muss. Und da bietet sich WebSockets an.

4.2 Emulation im Web

- Informationsaustausch zwischen Web-Browser und Emulator

4.3 Transport & Synchronisation der Spieldaten

- Server-Client-Modell, wo die Emulation auf einem Game-Server statt findet. Vor-/Nachteile

4.4 Kommunikation zwischen Browser und Controller-App

5

Lösungsansätze

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

5.1 Vorhandene Teillösungen

5.1.1 melody-jsnes

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

5.1.2 webtendo

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

5.1.3 retroarch-web

- Zu kompliziert, snes9x nur ein Emulator unter vielen
 - Nicht benutzerfreundlich

- Reagiert nicht auf Touch-Events

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

5.2 Lösungsansatz 1: Game-Server

Definition list

Is something people use sometimes.

Markdown in HTML

Does *not* work **very** well. Use HTML tags.

Pro Client-Seite sehr einfach und portabel (jsmpeg spielt den mpeg-ts stream [mpeg1/mp2] ab und fertig).

Pro Super Browser-Unterstützung. Habe keinen Browser gefunden, der den Stream nicht abgespielt hat.

Contra Minimaler Prototyp (mit statischem Video) hat nur funktioniert, in dem Audio und Video getrennt kodiert und zum jsmpeg-WebSocket-Relay geschickt wurde.

Contra Der Quellcode des Emulator(frontend)s müsste angepasst und erweitert werden:

Contra retroarch-ffmpeg unterstützt Live-Streaming und es sind auch verschiedenste Optionen konfigurierbar. Problem: mpeg1/mp2 wird nicht unterstützt, zumindest habe ich keine funktionierende Konfiguration gefunden. Für die Unterstüt-

zung von mpeg1/mp2 müsste also der retroarch-ffmpeg-core angepasst werden (viel C, wenig Kommentare). Ist prinzipiell machbar, aber nicht mit meiner C-Expertise, und nicht innerhalb des gegebenen Zeitrahmens.

Contra Auch schwierig: Die Inputs müssten von den Clients an retroarch geschickt werden. Hier müsste wieder das Emulator-Frontend (retroarch) angepasst werden, C/C++. Man hätte wahrscheinlich einen WebSocket-Server bauen können, der dann Nachrichten mit Payload=Controller-Input enthält und diese Nachrichten dann auf retroarch-Controller-Eingaben mappt und an das Emulator-Frontend weitergibt.

Contra ffmpeg, libav, Encodings, (spezielles) Muxing, A/V-Sync, Doku jeweils katastrophal

Noch zu testen: Paralleles Encoden von Audio/Video mit retroarch-ffmpeg. Das war für jsmpeg sowieso nötig.

5.2.1 Vorteile und Nachteile

5.3 Lösungsansatz 2: Emulation im Browser

- Warum sind die Smartphones mit dem “eigenen Bildschirm” verbunden, und nicht mit dem Host-PC?
- Minimierung des lokalen Input-Lags zwischen Anzeige und Steuerung

Tamarys WebRTC-Plugin

5.3.1 Vorteile und Nachteile

5.4 Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

6

Methodik

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

7

Konzeption

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

- Technologien nicht neu beschreiben, verweisen auf Kapitel 3
- Zeigen, wie diese zusammen spielen, um das Ziel zu erreichen

7.1 Entwurf der Benutzeroberfläche

7.2 Software-Architektur

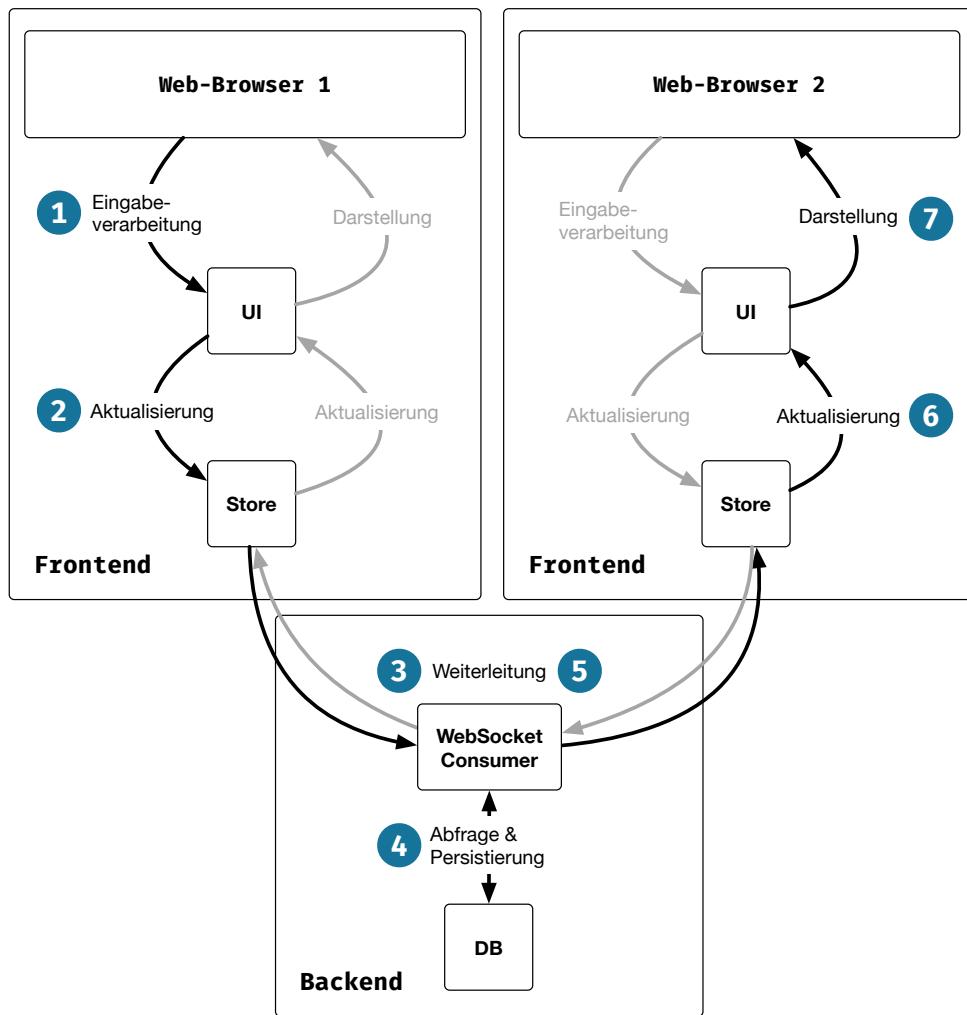


Abbildung 7.1: Bla foo bar.

8

Implementierung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

8.1 Entwicklungsumgebung

- Gitlab, Tickets, CI, etc.
- Heroku

8.2 Verwendete Software-Komponenten

- Ionic/Quasar: Macht Sinn, wenn *eine* App im Browser und mobil laufen muss.
Hier gibt es zwei alleinstehende Apps, die völlig verschiedene Dinge tun

8.3 Ausgewählte Implementierungsdetails

8.3.1 Cool #1

8.3.2 Cool #2

8.3.3 Integration von xnes/snes9x.js und vue.js

Drei notwendige Schritte:

1. Deaktivieren von eslint für Kompilierungsergebnis snes9x.js,
2. Webpack-Konfiguration: `node: {fs: 'empty'}`,
3. Anfügen von `export {Module as default}` ans Ende von snes9x.js, damit die Datei korrekt von Webpack erkannt wird.

8.4 Vollständigkeit

9

Evaluation

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Plan:

Große Fragen:

- Spielbar? User Tests?
- Nur im LAN, oder auch übers Internet?
- Welche Latenz war angestrebt? Erreicht?

Faktenbasiert, Metriken:

- Emulator-FPS
 - Emulator kann FPS anzeigen
 - Code so anpassen, dass die FPS durch JavaScript über eine API zugänglich sind und diesen Wert auslesen
 - Für den Test bietet sich Selenium an
 - Automatisiert
 - Mehrere Browser, Dienstleister: Saucelabs, BrowserStack
 - Schritte:
 - Match hosten
 - Spiel hochladen
 - Über N Zeiteinheiten FPS auslesen (bzgl. Statistik Tipps einholen)
- Netzwerk-Metriken von WebRTC
 - Automatisierung nicht ganz einfach, weil zwei Browser kommunizieren müssen. Unbekannt, ob Selenium das kann (also ein “Pair spawnen”, oder sowas)
 - Auslesen ist über das Control-Protokoll sehr gut machbar. Alle WebRTC-fähigen Browser haben dafür meist eine eigene Ansicht, auf der Debug-Informationen bzgl. WebRTC angezeigt werden

Weitere Ideen:

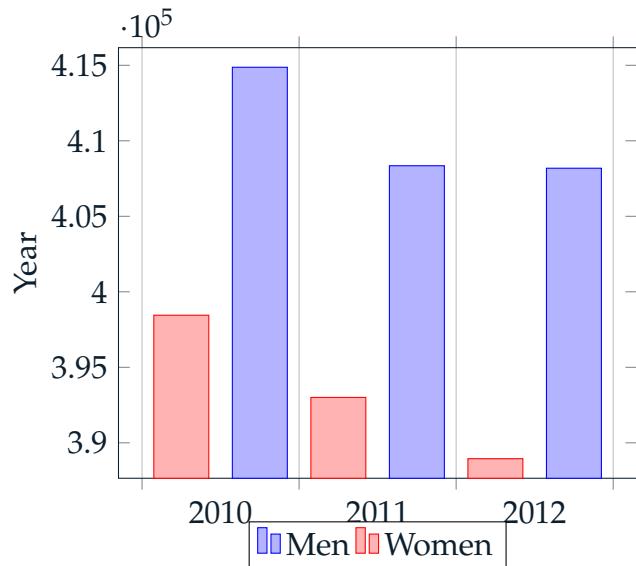
- WebRTC vs. WebSockets
- Wenn LAN-Spiel: Welchen Vorteil bringt WebRTC? (Wie viel weniger Latenz?)

- Emulator: emscripten-Backends vergleichen. asm.js vs. WebAssembly. Emulator funktioniert mit beidem in den zu unterstützenden Browsern (Chrome, Firefox). Bringt es Vorteile? Bessere Framerate? Wie viel besser? Andere Metriken?

9.1 Aufbau der Messumgebung

9.2 Ergebnisse und Beobachtungen

https://www.sharelatex.com/learn/Pgfplots_package



9.3 Diskussion und Bewertung

10

Fazit

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

10.1 Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

10.2 Bewertung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

10.3 Ausblick

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich

die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Verzeichnisse

Abbildungsverzeichnis

3.1 Home town flower	12
3.2 Controller-App übernimmt die Eingabe der Steuerbefehle	13
3.3 Home town Test	13
3.4 Cool	15
3.5 Ansichten und Navigationspfad der Controller-App.	20
7.1 Home town flower	36

Literatur

Print

Grigorik, Ilya. *High Performance Browser Networking: What every web developer should know about networking and web performance*. 1. Aufl. O'Reilly Media, Sep. 2013. ISBN: 9781449344764. URL: <https://hpbn.co>.

Krug, Steve. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability (3rd Edition) (Voices That Matter)*. 3. Aufl. New Riders, Jan. 2014. ISBN: 9780321965516. URL: <https://sensible.com/dmmt.html>.

Web

Burnett, Daniel C., Adam Bergkvist, Cullen Jennings, Anant Narayanan und Bernard Aboba. *Media Capture and Streams (W3C Candidate Recommendation)* 19 May 2016.

- Mai 2016. URL: <https://www.w3.org/TR/mediacapture-streams> (besucht am 21.07.2017).
- E. Rescorla, Ed., C. Jennings und J. Uberti. *JavaScript Session Establishment Protocol*. URL: <https://rtcweb-wg.github.io/jsep> (besucht am 16.07.2017).
- emulation-general.wikia.com. *Super Nintendo emulators*. URL: http://emulation-general.wikia.com/wiki/Super_Nintendo_emulators (besucht am 10.06.2017).
- Fablet, Youenn. *Announcing WebRTC and Media Capture*. URL: <https://webkit.org/blog/7726/announcing-webrtc-and-media-capture> (besucht am 08.06.2017).
- Inc., Apple. *What's New in Safari: Safari 11.0*. URL: https://developer.apple.com/library/content/releasenotes/General/WhatsNewInSafari/Safari_11_0/Safari_11_0.html (besucht am 08.06.2017).
- Mann, Ben. *Multiplayer JS game platform*. URL: <https://github.com/8enmann/webtendo> (besucht am 07.06.2017).
- Richards, Gregor. *Netplay core testing*. URL: <https://github.com/GregorR/RetroArch/wiki/Netplay-core-testing> (besucht am 10.06.2017) (siehe S. 3).
- Rouyrre, Maxime. *WebAssembly 101: a developer's first steps*. URL: <https://blog.openbloc.fr/webassembly-first-steps> (besucht am 01.07.2017).
- Wikipedia. *List of video game emulators*. URL: https://en.wikipedia.org/wiki/List_of_video_game_emulators#Super_NES (besucht am 10.06.2017) (siehe S. 3).

Index

Glossar

Accessibility

Canvas

Controller Gerät zur Steuerung von Computerspielen. Bei alten Konsolen über Kabel verbunden

Django Web-Framework für die Programmiersprache Python

Docker Plattform zur Erstellung und Verwaltung von Software-Containern

Emulator Computer-Programm zur Emulation von Konsolenspielen

Host Dienstanbietender Teilnehmer in einem Computer-Netzwerk

WebSockets , [RFC 6455](#)

Akronyme

GUI Graphical User Interface

HTTP Hyper Text Transfer Protocol, [RFC 2616](#)

MPEG-1 [Moving Picture Experts Group Phase 1](#)

NAT Network Address Translation, [RFC 1631](#)

NES Nintendo Entertainment System

ROM Read-Only Memory

SNES Super Nintendo Entertainment System

STUN Session Traversal Utilities for [NAT](#), [RFC 5389](#)

TURN Traversal Using Relays around [NAT](#), [RFC 5766](#)

WWW World Wide Web

Anhang

Entwürfe



DROP GAME ROM HERE

PICK A NAME

SELECT NO. OF PLAYERS

2 3 4

HOST GAME



GAME
Super Tennis

HOST
Alice

PLAYERS
1/2

Join this Game



GAME
Mario World

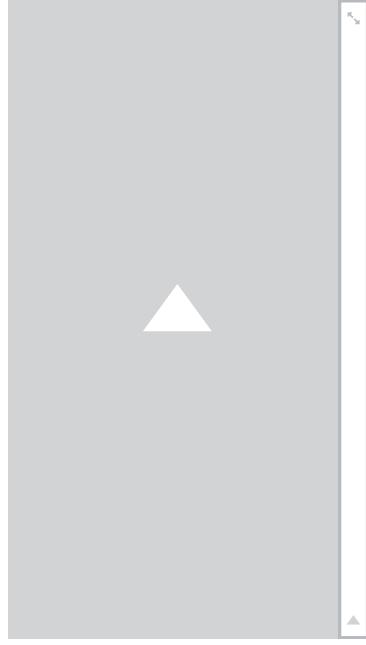
HOST
Bob

PLAYERS
2/2

Join this Game



Super Bomberman



Alice42: Lorem ipsum is simply dummy text of the printing and typesetting industry. Lorem ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

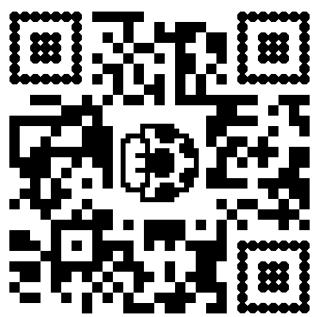
Write a Message

Send



Super Bomberma

Use your smartphone as a controller!
Just scan the code using the **SMES app**.



Alice42: Lorem ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Write a Message

Send



