Titelblatt

Abstract

- Ergebnisse aus den Spieletests, der Entwicklung und der Literaturrechereche zusammengefasst erläutern.
- Erst am Schluss schreiben!aa
- Wichtig für den ersten Eindruck

Contents

Chapter 1

Einleitung

Teil 1

- Beschreibung des Videospiels Palim-Palim (inkl. Screenshot)
- Aufstellung der Forschungsfragen und den Erkenntnissen

Teil 2

- Ausgangslage inkl. Forschungsstand
- Relevanz der Problemstellung

Teil 3

- Grobe Beschreibung der angewendeten Methodik
  - Spielentwicklung (Architektur, Technologien)
  - Spieletests (Methoden)

Teil 4

- Aufbau des Dokuments und Überleitung in den theoretischen Teil

Chapter 2

Umfeldanalyse und Zielgruppe

- Beschreibung des Umfelds / Anwendungsdomäne (betagte Personen und Kinder)

Chapter 3

Intergenerationelles Spielen

- Forschungsstand (Welche Methoden/Ansätze werden angewendet?)
- Bisherige Erkenntnisse zu intergenerationellem Spielen

Chapter 4

Videochats in Videospielen

- Forschungsstand (Welche Methoden/Ansätze werden angewendet?)
- Bisherige Erkenntnisse zu Videochats in Videospielen

Chapter 5

Aufstellung der Forschungsfragen

- Lücken der bisherigen Forschung
- Aufstellung der Forschungsfragen

Chapter 6

Methoden

- In Palim-Palim verwendete Methoden
  - Methode der Spieletest
  - Methode der Spieletest-Auswertung

Chapter 7

Spieletests und Resultate

7.1 Resultate

- Ergebnisse
- Beantwortung der aufgestellten Forschungsfragen

7.2 Planung und Durchführung der Spieletests

- Organisation der Spieletests (Aufbau, Ablauf, Testpersonen, Testszenarien, Testumgebung)
- Beobachtungen

Chapter 8

Gamedesign

8.1 Spielkonzept

8.1.1 Gameplay-Loops

8.2 Spielvarianten

8.3 Design und Usability

Chapter 9

Implementation

9.1 Architektur

9.2 Technologien

9.2.1 WebRTC

WebRTC [https://webrtc.org/] ist ein Open-Source-Projekt, das die Echtzeitkommunikation von Audio, Video und Daten in Web- und nativen Anwendungen ermöglicht. Die Technologie ist in allen modernen Browsern sowie auf nativen Clients für alle wichtigen Plattformen verfügbar. Dabei wird zwischen zwei Browsern eine Peer-To-Peer-Verbindung aufgebaut, worüber die Daten gestreamt werden.

Signaling

WebRTC verwendet die RTCPeerConnection [https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection] JavaScript API, um Streaming-Daten zwischen Browsern zu kommunizieren. Zusätzlich wird aber auch einen Mechanismus benötigt, um die Kommunikation zu koordinieren und Kontrollnachrichten zu senden. Dieser Prozess wird als *Signaling* bezeichnet. Signaling-Methoden und -protokolle sind von WebRTC nicht spezifiziert und können je nach Anwendungsfall entsprechend gewählt werden.
In diesem Projekt wurde Socket.IO für die Signalisierung verwendt, da es sich mit seinem integrierten Room-Konzept für eine Video-Chat-App anbietet. Es gibt aber viele Alternativen, welche sich teilwiese besser für produktive Applikationen eignen (https://bloggeek.me/siganling-protocol-webrtc/). Palim-Palim verwendet eine Node.js Server-Applikation (implementiert in server.js),

welche als Signaling-Server fungiert. Der Server hat dabei folgende zwei Aufgaben.

Erstens fungiert er als Message Relay. Dies wir benötigt, um verschiedene Informationen von und zu den Clients zu senden, damit diese untereinander eine WebRTC-Peer-Verbindung aufbauen können:

```
socket.on('message', function (message) {
  log('Got message: ', message);
  socket.broadcast.emit('message', message);
});
```

Zweitens verwaltet er alle WebRTC-Videochat-'Räume':

```
if (numClients === 0) {
  socket.join(room);
  socket.emit('created', room, socket.id);
} else if (numClients === 1) {
  socket.join(room);
  socket.emit('joined', room, socket.id);
  io.sockets.in(room).emit('ready');
} else { // max two clients
  socket.emit('full', room);
}
```

Wie im Code ersichtlich ist, erlaubt unsere Applikation maximal zwei Peers in einem Raum.
Zusätzlich werden in unserer Applikation die Positionen der Gameobjekte über den Server synchronisiert (TODO).

STUN und TURN

WebRTC ist grundsätzlich so konzipiert, dass es Peer-to-Peer funktioniert. Benutzer können sich also auf dem direktesten Weg verbinden. Die Technologie ist jedoch darauf ausgelegt, mit realen Netzwerken zurechtzukommen: Client-Anwendungen müssen NAT-Gateways und Firewalls überwinden, und Peer-to-Peer-Netzwerke benötigen Fallbacks, falls die direkte Verbindung ausfällt. Als Teil dieses Prozesses verwenden WebRTC-APIs sogenannte *STUN-Server*, um die IP-Adresse ihres Computers zu ermitteln, und *TURN-Server* (Traversal Using Relay NAT), welche als Relay-Server fungieren, falls die Peer-to-Peer-Kommunikation fehlschlägt.[https://webrtc.org/getting-started/turn-server]

Palim-Palim verwendet als STUN-Server öffentlich verfügbare Server von Google. Als TURN UDP- und TCP-Fallback-Verbindungen werden die Konfigurationen aus dem WebRTC-Sample-Projekt übernommen. Diese Server

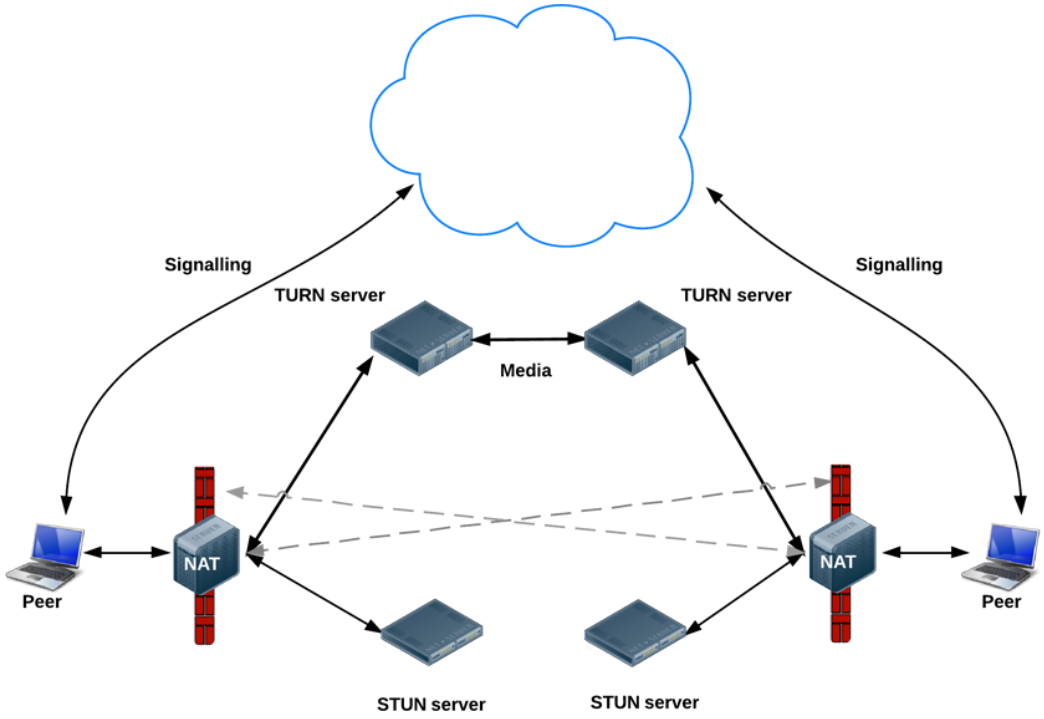

Figure 1: STUN, TURN, und Signalisierung in WebRTC. Quelle: https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/.

werden gratis angeboten und können mit Tools wie Google's Trickle ICE sample [https://webrtc.github.io/samples/src/content/peerconnection/trickle-ice/] auf ihre Verfügbarkeit getestet werden [https://bloggeek.me/webrtc-turn/]. Die STUN- und TURN-Server-Adressen müssen bei dem Aufbau der Peer-Connection vom Client an den Signaling-Server übermittelt werden. Dazu übermittelt jeder Palim-Palim-Client beim Verbindungsaufbau folgende pcConfig-Werte an den Server:

```
1       var pcConfig = {
2   'iceServers': [
3     {
4       'urls': 'stun:stun.l.google.com:19302'
5     },
6     {
7       'urls': 'turn:192.158.29.39:3478?transport=udp',
8       'credential': 'JZEOEt2V3Qb0y27GRntt2u2PAYA=',
9       'username': '28224511:1379330808'
10    },
11    {
12      'urls': 'turn:192.158.29.39:3478?transport=tcp',
13      'credential': 'JZEOEt2V3Qb0y27GRntt2u2PAYA=',
```

```
14        'username': '28224511:1379330808'
15      }
16    ]
17  };
```

In etwa 82 Prozent der Fälle ist die Peer-To-Peer-Verbindung stabil genug, und die TURN-Server werden überhaupt nicht benötigt. Jedoch müssen sie trotzdem bei jeder Verbinung angegeben werden. TODO more infos https://www.callstats.io/blog/2017/10/26/webrtc-product-turn-server

Sicherheit

Verschlüsselung ist für alle WebRTC-Komponenten obligatorisch, und seine JavaScript-APIs können nur von sicheren Quellen (HTTPS oder localhost) aus verwendet werden. Die Signalisierungsmechanismen sind allerdings nicht in den WebRTC-Standards definiert. Hier liegt es an den Entwicklern, sichere Protokolle zu verwenden. TODO¡Für unser Projekt noch definieren und ausformulieren¿

9.3 Sicherheit

9.4 Testing

9.5 Deployment und Betrieb (Anhang?)

Chapter 10

Fazit

- Zusammenfassung des Erreichten / Zielerreichung
- Zentrale Erkenntnisse
- Reflektion
- Mögliche Weiterentwicklungen (bezogen auf die Software)
- Weiterführende Forschung

Chapter 11

Literaturverzeichnis

Appendix A

Ehrlichkeitserklärung

Appendix B

Testprotokolle, weitere Spielkonzepte/Ideenfindung...