htw saar

Systems Neuroscience
& Neurotechnology Unit

VR Freifeld

Project Work

Saarland University of Applied Sciences
Faculty of Engineering

Submitted by	:	Dominik Limbach
Matriculation Number	:	3662306
Course of Study	:	Biomedical Engineering
First Supervisor	:	Prof. Dr. Dr. Daniel J. Strauss
Second Supervisor	:	Dr. Lars Haab

Homburg, January 2, 2018

Declaration

I hereby declare that I have authored this work independently, that I have not used other than the declared sources and resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. This work has neither been submitted to any audit institution nor been published in its current form.

Saarbrücken, January 2, 2018

Dominik Limbach

Contents

Declaration **3**

1 Introduction **5**

2 Problem Analysis and Goals **6**

3 Materials and Methods **7**
3.1 Versuchsaufbau 7
3.2 Versuchsablauf 10
3.2.1 Vorbereitung Proband/in 10
3.2.2 Ablauf 10
3.3 Code 11
3.3.1 Matlab 11
3.3.2 Unity 14

4 Discussion **20**

5 Conclusions and Future Work **21**

Bibliography **22**

1 Introduction

Der Präzedenzeffekt beschreibt die räumliche Zuordnung einer Schallquelle anhand der ersten Wellenfront die das Ohr erreicht. Dabei werden alle Folgeereignisse, die in einem gewissen Zeitbereich nach der ersten eintreffen ebenfalls der selben Richtung zugeordnet. Der Effekt tritt dann auf wenn zwei gleiche Schallereignisse mit einer Verzögerungszeit von 2-30 ms am Ohr ankommen und ist zudem abhängig vom Pegel der Folgeereignisse.[1][2] Solche Folgeereignisse sind zum großen Teil Reflexionen, die von den Wänden des Raumes zurückgeworfen werden. Reflexionen, die das Ohr nach einer zu langen Verzögerungszeit erreichen werden als Echo oder separates Schallereignis wahrgenommen. Aus diesem Grund werden akustische Messungen, welche die Lokalisation von Schallquellen betreffen, unter Freifeld-Bedingungen durchgeführt. Um diese Bedingungen zu realisieren bedarf es einer umfangreichen Schallisolation des Messraums. Dieser Umstand sorgt für zusätzliche Kosten und zu einer Beschränkung des Versuchsaufbaus. Die virtuelle Realität bietet eine kostengünstige Möglichkeit verschiedenste Messbedingungen zu simulieren und damit wissenschaftliche Experimente durchzuführen. Wie sich nun eine Freifeld-Umgebung zur Durchführung von akustischen Messungen in der virtuellen Realität realisieren lässt ist Thema dieser Projektarbeit.

[1] siehe [1],S.140ff
[2] siehe [2],S.103ff

2 Problem Analysis and Goals

Bei akustischen Experimenten unter Freifeldbedingungen erfolgt die Präsentation der akustischen Signale in der Regel über Lautsprecher. Das Signal wird von der Versuchsperson entsprechend der Position des Lautsprechers im Raum wahrgenommen. Im virtuellen Setup werden die akustischen Signale von einer virtuellen Quelle erzeugt, deren Position im virtuellen Raum der des realen Lautsprechers entspricht. Die Wiedergabe erfolgt jedoch über einen Kopfhörer. Generell scheint ein Ton , der von einem Kopfhörer erzeugt wird, seinen Ursprung im Kopf zu haben. Durch die Erzeugung eines virtuellen akustischen Raumes lässt sich die Anordnung der realen Schallquellen simulieren und somit eine räumliche Lokalisation der Schallquellen ermöglichen[1].

[1] siehe [3]

3 Materials and Methods

3.1 Versuchsaufbau

Die Grundlage des Versuchsaufbaus ist der virtuelle Raum in dem sich die Versuchsperson während der Durchführung aufhalten wird. Die Erzeugung des virtuellen Umfelds wurde mit der Unity Software, Version 5.6 von der Firma Unity Technologies durchgeführt.
Das Grundgerüst des Raumes wird durch einen Würfel, mit drei Metern Kantenlänge und einer Wandstärke von einem Zentimeter gebildet. Sämtliche Raumoberflächen wurden mit dem Standard Cube Mesh und einen Neutralen grauen Farbton versehen. Die Umgebung wird mittels vier identischer Lichtquellen beleuchtet. Diese sind vom Typ Spotlight und sind mit einem Winkel von plus neunzig Grad zu der Decke auf den Boden gerichtet. Um eine ausgeglichene Beleuchtung, ohne Überstrahlung, an den Kanten des Raumes zu gewährleisten befinden sich die Lichtquellen in einem Abstand von zwei Metern oberhalb der Raumdecke und ihre Intensität wurde auf achtzig Prozent eingestellt. Für den Benutzer werden die Lichtquellen durch vier Deckenlampen dargestellt.
Die Gestaltung wurde bewusst schlicht gehalten, da die Realitätsnähe nicht im Fokus des Experimentes liegt.
Das Setup für den Freifeldversuch, bestehend aus vier virtuellen Lautsprechern, einem virtuellen Mikrofon, einem visuellen Hinweis und einer zentralen Markierung für den Probanden auf dem Boden wurde im Mittelpunkt des virtuelle Raum implementiert.
Die Audioquellen, wurden in Form spezieller Game Objects innerhalb der Unity Umgebung realisiert. Diese sogenannten Audio Sources sind in der Lage einen beliebigen Audio Clip wiederzugeben[1].Um die Position jeder Audio Source im Raum erkennen zu können wurden diese jeweils mit dem 3D-Modell eines Lautsprechers verknüpft und auf einem kleinen Podest positioniert. Die vier Audioquellen befinden sich auf einer Kreisbahn um die Probanden Markierung mit einem Radius von zwei Metern.

[1] siehe [4]

Die Positionen der vier Quellen lauten wie folgt:

- Links Lateral: $-90°$
- Links Frontal: $-30°$
- Rechts Frontal: $+30°$
- Rechts Lateral: $+90°$

Der Abstand der virtuellen Lautsprecher zum Boden beträgt 1,20 Meter.
Die Audioquellen sind mithilfe des audio spatializer plugin, welches direkt in Unity implementiert ist, in der Lage die Wiedergabe des Audio Clip so zu beeinflussen, dass dieser seiner Position im virtuellen Raum entsprechend wahrgenommen werden kann.
Die Aktivierung dieser Funktion erfolgt über den Unity Audio Manager[2].
Dieser wurde wie folgt konfiguriert:

- Default Speaker Mode: Stereo
- System Sample Rate: 48000 Hz
- Spatializer Plugin: MS HRTF Spatializer

Die restlichen Einstellmöglichkeiten verbleiben auf den Standardwerten.
Darüber hinaus muss jede der erstellten Audioquellen entsprechend konfiguriert werden.

- Spatialize: On
- Volume: 1
- Stereo Pan: 0
- Spatial Blend: 1
- Spread: 0

An dieser Stelle wird dem Benutzer von Unity mittels der Volume Rolloff Einstellung die Möglichkeit geboten jede der oben genannten Parameter innerhalb des Einflussbereichs der Audioquelle, abhängig von der Entfernung des Hörers zur Quelle, zu beeinflussen.

[2] siehe [5]

Im Rahmen dieses Aufbaus wurde ein Custom Rolloff gewählt, welcher zu einer Dämpfung des Signals innerhalb eines Radius von acht Metern um die Quelle führt. Dies kann dem Probanden während des Versuch zusätzliche Informationen über seine Entfernung zu der jeweiligen Quelle geben. Die restlichen, der oben genannten Parameter, bleiben innerhalb des gesamten Einflussbereichs der Quelle unverändert.
Der von den Audioquellen abgespielte Audio Clip wird von einem virtuellen Mikrofon aufgezeichnet. Dieses Mikrofon ist in Form des Audio Listeners in der Unity Engine implementiert und ist nicht konfigurierbar. Das Mikrofon wurde als eine Komponente der Kopfkamera, der sogenannten camera(ears), angelegt. Somit wird jede Ausgabe der Audioquellen aus der Sicht des Trägers der VR-Brille wahrgenommen.
Der visuelle Hinweis wurde in Form eines grünen Pfeils implementiert. Dieser befindet sich an der Wand des virtuellen Raumes, welche sich hinter den Audioquellen befindet. Seine Position entspricht 0° auf der Kreisbahn und sein Abstand zum Boden beträgt 1,20 Meter.

Abschließend wurde das HTC Vive kalibriert, der Untersuchungsraum vermessen und ein Stuhl auf der Markierung platziert.

3.2 Versuchsablauf

3.2.1 Vorbereitung Proband/in

Nachdem die Versuchsperson über den Ablauf informiert wurde wird sie angewiesen sich zur Markierung zu begeben, Platz zu nehmen, die VR-Brille mit dem Kopfhörer aufzusetzen und entsprechen zu fixieren.

3.2.2 Ablauf

Zu Beginn des Experiments werden die Untersuchungsparameter durch den Versuchsleiter eingegeben. Über ein Matlab Programm, welches auch deren Eingabe verwaltet, wird das Wiedergabeprotokoll erstellt, die Untersuchungsdaten gespeichert und an Unity übergeben. Das Wiedergabeprotokoll wird entsprechend der eingegebenen Parameter automatisch erstellt. Für jedes wiederzugebende Signal wird eine zufällige Quelle zur Wiedergabe bestimmt.
Sobald das Protokoll erstellt und and Unity gesendet worden ist kann das Experiment gestartet werden.
Während des Experiments wird der Versuchsperson eine Folge von akustischen Signalen präsentiert. Der/Die Proband/in wird angewiesen nach jeder Signalwiedergabe die Richtung aus der das Signal wahrgenommen wurde anzugeben. Nachdem das Programm gestartet wurde wird der ausgewählte Audio Clip in einem fünf Sekunden Intervall von jeweils einer der vier Audioquellen wiedergegeben. Zu jeder Wiedergabe wird der Versuchsperson ein visueller Hinweis in Form einer Richtungsangabe durch einen grünen Pfeil in ihrem Sichtfeld dargeboten. Das Programm ist beendet, wenn die gewünschte Signalanzahl erreicht wurde.

3.3 Code

3.3.1 Matlab

```
% filename: freifeld.m
% Saarland University of Applied Sciences
% author: Dominik Limbach
% date: 01.11.2017
% description: program that reads the connection information
% from the target pc, establishes a tcp/ip connection, requests
% input (user data,exam parameter), creates a exam protokoll
% and then saves the data before sending it to the destination

clc;
clear;

% acquire connection information
txt = textread('configFF.txt','%s','delimiter','\n');
ipAdress = txt{1,1};
portNumber = str2num(['uint32(',txt{2,1},')']);

% create tcp object, set Port,assign Networkrole Client
tcpIpClient = tcpip(ipAdress,portNumber,'NetworkRole','Client')
    ;


% input messages
lastname_prompt ='Enter a last name. \n';
name_prompt = 'Enter a name. \n';
birthdate_prompt = 'Enter a birth date. (DD.MM:YYYY) \n';
examdate_prompt = 'Enter a date. (DD.MM:YYYY) \n';
arrow_prompt = 'Do you want the signal position and the sign
    position to match? (yes = true , no = false)\n';
signal_prompt = 'How many repetitions would you like to run?
    Please enter integer value.\n';
Error_prompt = 'Wrong input.Please enter a value of the correct
     type.\n';

% number of available Speakerpositions
speakerPositions = 4;

% input: user information
last_name = input(lastname_prompt,'s');
```

```
36 name = input(name_prompt,'s');
37 date_birth = input(birthdate_prompt,'s');
38 date_exam = input(examdate_prompt,'s');
39
40 % break variable for the input check
41 ok = false;
42
43 while ok == false
44 % linked is the variable which defines the paradigm
45 % linked = true => audio position and sign position match
46 % linked = false => audio position and sign position dont match
47 linked = input(arrow_prompt);
48 if isa(linked,'logical') == true
49     ok = true;
50 else
51    disp(Error_prompt)
52    disp(' ')
53    ok = false;
54 end
55 end
56
57 % break variable for the input check
58 ok = false;
59
60 while ok == false
61 % number of Signals per Set
62 numberSignals = input(signal_prompt);
63 if isa(numberSignals,'numeric') && (numberSignals >= 1) == true
64     ok = true;
65 else
66    disp(Error_prompt)
67    disp(' ')
68    ok = false;
69 end
70 end
71
72 % initialize position arrays
73 arrowPosition = zeros(1,numberSignals);
74 audioPosition = zeros(1,numberSignals);
75 counter = 1;
76
77 while counter <= numberSignals
78
79 if linked == true
```

```
80      audioPosition(1,counter) = randi([0,(speakerPositions-1)
            ],1,1);
81      arrowPosition(1,counter) = audioPosition(1,counter);
82      counter = counter+1;
83  else
84      audioPosition(1,counter) = randi([0,(speakerPositions-1)
            ],1,1);
85      arrowPosition(1,counter) = randi([0,(speakerPositions-1)
            ],1,1);
86      counter = counter+1;
87  end
88  end
89
90  %save user_file
91  filename = 'auditoryExam.mat';
92  save(filename,'last_name','name','date_birth','date_exam','
        numberSignals',...
93  'linked','audioPosition','signPosition');
94
95  % prepare data to send
96  data = [numberSignals ,audioPosition ,arrowPosition];
97  % open tcp object, send to target and close tcp object
98  fopen(tcpIpClient);
99  fwrite(tcpIpClient,data)
100 fclose(tcpIpClient);
101 % end of program
```

3.3.2 Unity

```
// filename: UnityServer.cs
// Saarland University of Applied Sciences
// author: Dominik Limbach
// date: 01.11.2017

// description: the program will read data from the stream and store
// crucial information into variables


using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Net.Sockets;
using System.Net.NetworkInformation;
using UnityEngine;
using System.Net;
using System;

public class UnityServer : MonoBehaviour
{
    // Use this for initialization
    TcpListener listener;
    String msg;
    // determines data size
    byte[] data = new Byte[256];
    // intialize exam parameter variables
    public int numberSignals;
    public Int32[] audioPositions;
    public Int32[] arrowPositions;

    private void Awake()
    {
        numberSignals = 5;
        audioPositions = new Int32[numberSignals];
        arrowPositions = new Int32[numberSignals];
    }

    void Start()
    {
        listener = new TcpListener(8633);
        listener.Start();
```

```
42          print("Server is listening.");
43
44      }
45
46      // Update is called once per frame
47      void Update()
48      {
49          if (!listener.Pending())
50          {
51          }
52          else
53          {
54              print("Data received.");
55              TcpClient client = listener.AcceptTcpClient();
56              NetworkStream stream = client.GetStream();
57              StreamReader reader = new StreamReader(stream);
58              msg = reader.ReadToEnd();
59
60              // Store the received Data in data array
61              data = System.Text.Encoding.ASCII.GetBytes(msg);
62
63              // extract the number of signals per set
64              numberSignals = Convert.ToInt32(data[0]);
65
66              audioPositions = new Int32[numberSignals];
67              arrowPositions = new Int32[numberSignals];
68
69              // extract data from stream
70              for (int i = 0; i < numberSignals; i++)
71              {
72                  // extract the audio positions
73                  audioPositions[i] = Convert.ToInt32(data[i
                        + 1]);
74                  // extract the arrow positions
75                  arrowPositions[i] = Convert.ToInt32(data[i
                        + 1 + numberSignals]);
76              }
77
78          }
79      }
80  }
```

```
1  // filename: ArrayControl.cs
2  // Saarland University of Applied Sciences
3  // author: Dominik Limbach
4  // date: 01.11.2017
5
6  // description: the program controls the activation of
       both arrow and audio sources
7  // it will gather the protokoll information from the
       UnityServer script
8  // and execute the protokoll
9
10
11
12 using System;
13 using System.Collections;
14 using System.Collections.Generic;
15 using UnityEngine;
16
17 public class ArrayControl : MonoBehaviour
18 {
19     // Use this for initialization
20     public AudioSource[] audioSource; // The array to
           contain the audiosources
21     public Vector3 arrowRotation; // Vector3 to control the
           arrow rotation
22     public float rotX; // Float variable which determines
           the angle of rotation on the Y-axis
23     public float[] angles; // Array that holds possible
           angles for the arrow rotation
24     public AudioClip sound; // Variable to hold the
           SoundClip
25     public Int32[] arrayAudio; // Array to import the Audio
           Position order in to
26     public Int32[] arrayArrow; // Array to impoert the
           Arrow Position Order in to
27     public int arraySize; // Defines size of arrayAudio and
           arrayArrow according to the imported numberSignals
28     private bool beeingHandled; // Bool for the subroutine
29     public int numberSpeakers; // Number of speakers
           available
30     public GameObject scriptObject; UnityServer
           getAudioPositions, getArrowPositions,
           getNumberSignals; // Variables for imported Variables
31
```

```
32      void Awake()
33      {
34          getAudioPositions =
               scriptObject.GetComponent<UnityServer>();
35          getArrowPositions =
               scriptObject.GetComponent<UnityServer>();
36          getNumberSignals =
               scriptObject.GetComponent<UnityServer>();
37      }
38
39      void Start()
40      {
41          // Load the five audiosources into the audioSource
               array
42          audioSource =
               GetComponentsInChildren<AudioSource>();
43          // Initialize the angles array for arrow positioning
44          angles = new float[5];
45          angles[0] = 90.0f;
46          angles[1] = 45.0f;
47          angles[2] = 0.0f;
48          angles[3] = -45.0f;
49          angles[4] = -90.0f;
50
51          rotX = angles[2];
52          beeingHandled = false;
53
54      }
55      // Update is called once per frame
56      void Update()
57      {
58          arraySize = getNumberSignals.numberSignals;
59          arrayArrow = new Int32[arraySize];
60          arrayAudio = new Int32[arraySize];
61
62          for (int i = 0; i < arraySize; i++)
63          {
64              arrayAudio[i] =
                   getAudioPositions.audioPositions[i];
65              arrayArrow[i] =
                   getArrowPositions.arrowPositions[i];
66          }
67
68          if (Input.GetKey(KeyCode.Space) && beeingHandled ==
```

```
            false)
69          {
70              StartCoroutine(HandleIt());
71          }
72      }
73
74      private IEnumerator HandleIt()
75      {
76          beeingHandled = true;
77
78          for (int i = 0; i < arraySize; i++)
79          {
80              if (arrayAudio[i] == 0)
81              {
82                  if(arrayArrow[i] == arrayAudio[i])
83                  {
84                      rotX = angles[0];
85                      print("Audio from L90 , Arrow to L90");
86                  }
87                  else
88                  {
89                      int x = arrayArrow[i];
90                      rotX = angles[x];
91                      print("Audio from L90");
92                  }
93                  audioSource[0].PlayOneShot(sound);
94              }
95              if (arrayAudio[i] == 1)
96              {
97                  if (arrayArrow[i] == arrayAudio[i])
98                  {
99                      rotX = angles[1];
100                     print("Audio from L30 , Arrow to L30");
101                 }
102                 else
103                 {
104                     int x = arrayArrow[i];
105                     rotX = angles[x];
106                     print("Audio from L30");
107                 }
108                 audioSource[1].PlayOneShot(sound);
109             }
110             if (arrayAudio[i] == 2)
111             {
```

```
112                     if (arrayArrow[i] == arrayAudio[i])
113                     {
114                         rotX = angles[3];
115                         print("Audio from R30 , Arrow to R30");
116                     }
117                     else
118                     {
119                         int x = arrayArrow[i];
120                         rotX = angles[x];
121                         print("Audio from R30");
122                     }
123                     audioSource[2].PlayOneShot(sound);
124                 }
125                 if (arrayAudio[i] == 3)
126                 {
127                     if (arrayArrow[i] == arrayAudio[i])
128                     {
129                         rotX = angles[4];
130                         print("Audio from R90 , Arrow to R90");
131                     }
132                     else
133                     {
134                         int x = arrayArrow[i];
135                         rotX = angles[x];
136                         print("Audio from R90");
137                     }
138                     audioSource[3].PlayOneShot(sound);
139                 }
140 
141                 yield return new WaitForSeconds(5.0f);
142             }
143 
144             beeingHandled = false;
145             print("Routine finished!");
146         }
147 
148 
149 }
```

4 Discussion

Da das System nicht an Versuchspersonen getestet wurde lässt sich die Fähigkeit des Systems, die akustischen Eigenschaften eines Freifeldes zu simulieren, nur subjektiv bewerten. Zwar ist die Leistung ausreichend um eine räumliche Lokalisation der Quellen im virtuellen Raum zu ermöglichen, jedoch ist keine Aussage über deren Präzision möglich. Ein direkter Qualitätsvergleich zwischen VR-Setup und einem Freifeld lässt sich mit diesem Versuchsaufbau nicht durchführen und ist folglich nicht untersucht worden.
Ein entsprechender Vergleich wurde jedoch schon von Griffin D. Romigh, Douglas S. Brungart und Brian D. Simpson durchgeführt [1]. Hier wurde ein Versuchsaufbau verwendet der es ermöglicht die Versuchsperson unter beiden Bedingungen zu testen und somit einen objektiven Vergleich zu ziehen.
Während der Entwicklung wurde das Experiment zusätzlich mit unterschiedlichen Audio-Ausgabegeräten durchgeführt. Dies führte zu Verbesserungen in der räumlichen Wahrnehmung. Die folgenden Varianten sind nach ihrer Qualität in steigender Reihenfolge angeordnet.

1. **HyperX Cloud II:** Kopfhörer, Over-Ear Headset, geschlossen

2. **Audio-Technica ATH-AD900X:** High-Fidelity Kopfhörer, offen

3. **FX Audio DAC-X6:** DAC/AMP , in Kombination mit den oben genannten Kopfhörern

Möglichkeit Nummer drei bietet zwar die beste Qualität aufgrund eines höheren Signal-Rausch-Abstandes ($>= 105dB$) sowie einer besseren Kanalseparierung, bedarf jedoch zusätzlicher Adapter um das HTC-Vive mit einem DAC zu verbinden. Die beschränkte Länge von gängigen Adaptern(3.5mm Klinke auf USB) hat sich in Kombination mit der hier verwendeten Hardware als problematisch erwiesen.

[1] siehe [6]

5 Conclusions and Future Work

Abschließend lässt sich sagen, dass mit einem VR-Setup zwar zufriedenstellende Ergebnisse zu erzielen sind. In der Realität ist es jedoch sehr schwer die Qualität eines konventionellen Aufbaus zu erreichen. Bei der Entscheidung für ein VR-Setup spielen vor allem niedrige Kosten, Ortsungebundenheit und Vielseitigkeit eine tragende Rolle.
Bestehen hohe Qualitätsansprüche bezüglich der räumlichen Lokalisation ist nach wie vor ein Freifeld Setup mit entsprechender Hardware zu bevorzugen.

Bibliography

[1] A. Friesecke. *Die Audio-Enzyklopädie: Ein Nachschlagewerk für Tontechniker*. De Gruyter Reference. De Gruyter, 2014. ISBN 9783110340181. URL https://books.google.de/books?id=iqboBQAAQBAJ.

[2] S. Weinzierl. *Handbuch der Audiotechnik*. VDI-Buch. Springer Berlin Heidelberg, 2009. ISBN 9783540343011. URL https://books.google.de/books?id=Lf0mBAAAQBAJ.

[3] Zhan H. Zhou and Edward S. Rogers. Sound localization and virtual auditory space.

[4] Unity user manual (2017.2), 2017. URL https://docs.unity3d.com/Manual/class-AudioSource.html.

[5] Spatial sound in unity, 2017. URL https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_sound_in_unity.

[6] Griffin D. Romigh, Douglas S. Brungart, and Brian D. Simpson. Free-field localization performance with a head-tracked virtual auditory display. *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING*, 2015.