TH Köln

PROJECT REPORT

Implementing spatialized auditory training in VR: A proof of concept on the viability of Unity as a platform for a Speech Reception Threshold training application

Alexander Müller

Supervised by
M.Sc. Melissa Andrea Ramírez Caro
&
Prof. Dr. Christoph Pörschmann

February 3, 2022

Abstract

Central Auditory Processing Disorder (CAPD) is described by the American Speech-Language-Hearing Association (ASHA) as a condition, which "may lead to or be associated with difficulties in higher order language, learning, and communication functions" without being caused by actual hearing loss or inabilities [1]. Focusing on the aspect of spatial hearing Cameron and Dillon established the term Spatial Processing Disorder (SPD) and designed the Listening in Spatialized Noise (LiSN) & Learn auditory training software to improve binaural processing abilities of affected children [2].

Based on this foundation a new training program was developed as an open-source project with Virtual Reality (VR) support inside the *Unity* game development engine. Apart from offering a free alternative to the follow up product of the original LiSN & Learn software (*Soundstorm* [3]), this project makes initial steps towards evaluating whether *Unity* is a viable platform for auditory training applications. In further iterations, potential improvements to the concept may be reviewed, such as the inclusion of head tracking.

Contents

1 Introduction

Spatial hearing describes the ability to localize the origin of a given sound event by using binaural and spectral cues of the audio signal. This is typically associated with being useful for orientating or reacting to events that are currently outside the field of view.

But spatial hearing also allows to separate different sound sources from each other, and therefore helps to manage noisy environments. This is especially important if the incoming audio signals are similar to each other [4]. A popular example for this is the *cocktail-party-effect*, which describes a scenario where a listener is challenged by understanding a single speech signal in an environment with many simultaneous speakers. It has been found that listeners with hearing impairments face significantly more problems in such a situation than listeners with normal hearing. One of the main reasons for this difference has been identified as the ability to use spatial separation to distinguish the audio sources [5][6][7].

Alongside being an issue of sensorineural hearing impairment, it has been found that difficulties in understanding speech against background noise are also a common symptom of CAPD [8]. Focusing on the aspect of spatial hearing, Cameron and Dillon defined the term Spatial Processing Disorder (SPD) as "a condition whereby individuals are deficient in their ability to use binaural cues to selectively attend to sounds arriving from one direction while simultaneously suppressing sounds arriving from another" [2].

The concept of using training to mediate the effects of auditory processing disorders has already been the subject of research many times over the last decades. Some examples for this are projects like *Phonomena*, *Earobics* or *Fast ForWard*. However, none of these examples provided spatial separation between the different stimuli. Using this shortcoming as a starting point, Tyler et al. created a new training program that challenged the user with spatially separated speech in noise [9]. Building up on this foundation and their previous work on projects like the Listening in Spatialized Noise - Sentences Test (LiSN-S), Cameron and Dillon designed the LiSN & Learn auditory training software. Inside a publication in the *Journal of the American Academy of Audiology* [2], the new training program is elaborated and the results of a small study are presented. To do an initial verification of the efficacy of LiSN & Learn, a small group of participants - who have previously been diagnosed with SPD via LiSN-S - is set to use the new training software over a period of three months.

Over the course of this project the LiSN & Learn software will be reworked as an open-source VR application based on Unity. The combination of 3D environments, VR peripherals and advanced 3D audio is expected to allow the establishment of even more complex and realistic training options and high user engagement.

1.1 Method

In order to establish a foundation on which the efficacy of Unity as a platform for the implementation of auditory training applications can be evaluated, the LiSN & Learn program from Cameron and Dillon will be recreated. To ensure that the new iteration includes all important aspects of the original and therefore allows a comparison between both programs, an extensive analysis of LiSN & Learn has to be done. Furthermore, the design of the new project shall incorporate the aim of establishing Unity as platform. For this purpose - and to provide an orientation for the development process - a concept of the new application will be created. Special care has to be taken towards the flexibility and extensibility of the project to establish a code-base, from which the implementation of similar applications can be realized in a faster and more efficient manner.

1.2 Motivation

Re-creating the LiSN & Learn training software as an open-source VR application offers several interesting perspectives and advantages. Apart from establishing an initial reference point for future efficacy evaluations, the new framework also allows for the fast implementation of complex features.

In addition to taking advantage of the given utility functions included in Unity, the support of VR peripherals offers a whole new range of opportunities. The idea of using VR to establish highly immersive frameworks for educational and training purposes to improve student motivation and the quality of learning materials has been the subject of research multiple times [10]. There is, however still a great difference between the expected usage of this technology in the 1990s and its actual deployment in learning situations. This topic has been discussed in detail by Kavanagh et al. [11]. Considering the case of auditory training, the addition of matching visual stimuli might be very advantageous, since human perception naturally relies on multiple streams of information [12].

Another interesting aspect is the inclusion of mobility into the auditory scene. Due to the available audio spatializers, it's relatively easy to add moving sound sources with correct 3D audio rendering into the training process. This allows the inclusion of new metrics - like the Minimum Audible Movement Angle (MAMA) - into the system. On a similar note, the head tracking capabilities of VR peripherals also offer the opportunity to include the usage of dynamic spatial cues created by head movements to resolve front-back confusions [13].

Developing the new training game as an open-source project not only allows free distribution but might also lead to other researchers adapting and improving the foundation that is to be created here. However, this approach also has some drawbacks when it comes to the inclusion of third-party assets, since their license and usage conditions have to be considered.

Combining open-source distribution with an easy setup in both hardware and software,

the new project aims to also increase the overall accessibility of the training program. A more wide-spread use of auditory training to combat SPD would not only be beneficial for affected children, but also holds the option to collect more data for future research.

2 Fundamentals

In this section, some underlying principles and theoretical foundations of both the old and the new auditory training program will be discussed. The following descriptions are not meant to be seen as a thorough explanation, but rather a brief introduction to the topic.

2.1 Spatial hearing

As described in-depth by Jens Blauert [14], humans can use several cues to localize the origin of a given sound event. Taking advantage of having two ears - and therefore two points at which a sound signal can be registered - a listener can use interaural differences to determine the direction from which a noise originated. If a sound wave arrives from the right side, the head of the listener will cause a *shadowing effect*, resulting in an attenuation of the sound towards the left ear. The resulting variation is referred to as Interaural Level Difference (ILD) and changes alongside the azimuth angle from which the sound originated. Additionally, the length of the paths towards the ears also depends on the direction from which the signal reaches the listener. This causes a time deviation at which the sound is registered on each ear. This effect is described as Interaural Time Difference (ITD) and is especially important for low-frequency noises, which are not affected as much by the *head shadowing* effect and therefore are harder to locate using level differences alone[1].

While interaural differences are very useful for determining the azimuth angle of a noise source, they are lacking when it comes to elevation (see *cones of confusion*). Based upon the path a sound wave has to take around the listener's head, upper body, and outer ears, the spectrum of the original signal is altered. This effect can be described as a directional filter. However, a listener can only take advantage of these spectral coloring effects if the association between direction and alteration of the signal is already known.

A more detailed look into the subject is given by Stecker and Gallun in their educational work on binaural hearing [15].

2.2 3D audio

The concept of 3D audio can broadly be described as a process that allows to add virtual spatial information to an audio stream. As already discussed in the previous section, humans rely on a variety of cues to localize the source of a given noise. The effects caused by the path which the sound wave has to travel from its source to the listeners' ears can

[1] As already mentioned, this section is only supposed to give a slight overview. Therefore, information like the extent to which ITD and ILD contribute to spatial hearing on a given set of frequencies will be left out.

be described collectively as Head Related Transfer Function (HRTF). These transfer functions can either be measured individually - taking into account distinct characteristics of the pinnas, ear channels etc - or by using a general approach (e.g. measuring a dummy-head[2]). Even though individualized HRTFs allow for more precise localization, it has been found that, for many purposes, non-individualized HRTFs also work sufficiently well [17]. These drawbacks can be mitigated further if the auditory stimuli are combined with correctly placed visual representations [18].

When embedding sound into virtual environments, not only aspects of human localization have to be considered but also sound propagation, including reflections, absorption, etc. [19]. To realize accurate spatial sound simulations, both hardware and software must fulfill a certain set of requirements. For example, headphones are often the preferred playback device since they provide very small cross-talk between the left and right ear and are not affected by room acoustics.

2.3 Auditory Training

Auditory training is used to mitigate the effects of processing disorders, the usage of hearing aids [20], cochlea implants [21] or similar means. In many cases, the training program is specifically designed to address a single processing issue that is usually associated with a certain demographic (e.g. adults with hearing impairments). The LiSN & Learn project from Cameron and Dillon provides a good example for this. The program aims to improve the Speech Reception Threshold (SRT) for school-aged children that have been diagnosed with SPD. SRT is defined by the researchers as the average Signal to Noise Ratio (SNR) at which target sentences could still be understood against spatially separated distractor speech. Listeners without any hearing impairments or processing disorders can take advantage of the binaural differences between the speech signals and therefore achieve better SRT performances than would be possible in a non-spatialized scenario[3].

2.4 Training capabilities

Finally, it is sensible to take a look at possible improvements through auditory training. In the initial publication discussing the LiSN & Learn software, Cameron and Dillon already presented some data on SRT improvements that could be achieved over the course of a 3 month long training period [2]. In this study, children between the ages of 6 and 11 years who have shown "abnormal auditory behavior relative to their peers" and could be diagnosed with SPD through LiSN-S were selected as participants. With a sample group of 9 participants, an average gain of 9.7 dB in SRT levels was achieved in the LiSN & Learn software. Furthermore, LiSN-S performances before and after showed improvements as well, even though not as significant as those within the training software. After a period of 3 months without training, 8 out of 9 participants could still achieve similar LiSN-S results.

[2] A good example for this approach are the KEMAR measurements by the MIT [16]

[3] More information on the usage of binaural cues to improve listening performances in noisy environments can be found under the subject of *Spatial Release from Masking* [22].

Extending upon the data presented in the initial publication, Cameron and Dillon conducted a randomized blinded controlled study that compared improvements in LiSN-S performance between a LiSN & Learn group and a control group training with the *Earobics* program[4]. In total, 10 suitable participants trained over the course of 12 weeks. The 5 members of the LiSN group showed an average gain of 10.9 dB in SRT over the course of the training. The study concludes that an emphasis on spatialized sound is more suitable for re-mediating SPD than regular auditory training. As stated by the researchers "It was found that only the LiSN & Learn group showed significant improvements on the spatially separated conditions of the LiSN-S posttraining". According to both parents and teachers, the children of the LiSN & Learn group also showed improved real-world listening performances.

3 Approach

Before introducing a general concept for the new application in section 4, the requirements, goals and guidelines that have to be followed will be discussed here. Obviously, the actual development process will - at least to some degree - be dynamic and therefore some changes to the ideas presented here are to be expected. So instead of a strict requirement specification, this part shall rather be used as an aid for orientation and decision making over the course of this project.

3.1 Design principles

There are three main objectives which will be used as guidelines for this project:

1. Re-creating the original software to ensure comparability
2. Limiting use of third-party assets to free-license / open-source options
3. Design scripts / components to allow easy adjustments and extensions

Comparability To validate whether the new application does actually qualify as a training program to improve spatial processing abilities, it is sensible to re-create the existing software as closely as possible. This would not only allow to compare data from existing research with newly collected data, but also eliminate a lot of effort to check if, e.g. the chosen speech stimuli are appropriate for the desired purpose.

Free distribution As briefly mentioned in section 1.2, a free distribution of the training program holds several opportunities. First and foremost, eliminating licensing fees can help providing demographics access to the applications that could not afford the commercially achievable options. Additionally, publishing the source code allows other researchers and developers to adapt the concept to their individual requirements (e.g. translating the project into a different language). Assuming that these factors do indeed lead to an increased usage of the training program, there might be a new option to collect more data for future research. However, these aspects come with some drawbacks. For example,

[4] Apart from the insights given in [2] and [23] more information about Earobics can be found here [24]

open-source distribution requires special care when it comes to the inclusion of 3rd party assets to avoid violating license conditions. Fortunately, due to the widespread use of Unity for smaller applications, there are plenty of options available that often only require a mention of the original creator to be used and published.

Extensibility Writing flexible software usually takes significantly more effort during development, especially if the scope of the application is rather small. However, designing a project to be easily adjustable and extensible does not only establish the foundation for a successful open-source project, but is also useful when changes have to be made over the course of development. Ideally, a good compromise between additional time investments and potential benefits should be found during conceptualization.

3.2 Development tools

As already mentioned, the major part of the development will be done in Unity with the addition of Visual Studio as Integrated Development Environment (IDE). To include the VR functionalities, the Oculus SDK will be used. All audio editing is planned to be done using Audacity.

Unity overview Unity is a development platform that is mainly used as an engine for video games. It is free to use for non-commercial, educational, or small revenue purposes, and therefore qualifies as a foundation for an open-source project [25]. One of the key advantages of using such a framework is that issues like rendering of graphics and audio, portability between different systems etc. are already taken care of. Also, Unity includes a wide variety of features, components, and available documentation, which significantly speeds up the development process. Additionally, the Unity inspector - if used correctly - allows for several ways of adjusting parameters or switching assets without requiring any programming skills. This is especially interesting for this project, since it offers a low entrance barrier for other developers or researchers, that want to make small changes to the project to better fit their requirements. However, the vast scope of options given in Unity also adds some drawbacks, like an increased risk of choosing a sub-optimal approach when implementing features.

Oculus SDK The *Oculus SDK* offers a lot of utilities, which make the implementation of VR applications a lot easier. This goes from starting points for special camera rigs, to an adjusted input system, over to the *Oculus Audio Spatializer*. For the purpose of this project, the topic of audio spatialization is especially important. In the Oculus Developer Guide [26], the included features, important considerations, and additional guides are presented.

3.3 LiSN & Learn - Training Game

As mentioned before, the new application will be based strongly on the original LiSN & Learn software. In particular, the design of the *Training Game* should be matched precisely to ensure that the new game will remain comparable to the original. For this purpose, all major components of the *Training Game* will be presented in the following.

Setup The listener is challenged with understanding *target sentences* against background noise, consisting of two distractors talking with the same voice as the *target speaker*. The *target sentences* will be presented at a 0 degree azimuth angle, while the *distractor stories* will be shifted 90 degrees to the left or right, respectively (as shown in figure 1).

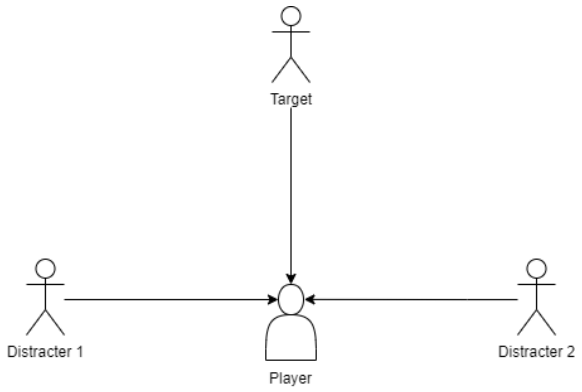


Figure 1 Setup

Task The player has to identify a randomly selected word from the previously played *target sentence* from a selection of **4 options**. Alternatively, there's also an *unsure* button, which leads to the current sentence being repeated **once**.

Adaptive SNR Based upon whether the participant guesses a word correctly (*hit*), incorrectly (*miss)* or selects the *unsure* option, the SNR between the target and distractor stimuli will be adjusted:

- Hit: - 1.5 dB SNR
- Miss: + 2.5 dB SNR
- Unsure: + 1.5 dB SNR

Practice mode To find the current SRT of the participant, a *practice mode* will be started at the beginning of each session. During this period, the SNR will be decreased by 3 dB each round. A **minimum of 5 practice rounds** has to be done before starting the actual training game. Once this minimum has been reached, the first incorrect or *unsure* response after a correct answer will end the *practice mode*.

Rewards When **5 consecutive correct answers** have been given, the participant is granted a reward sticker, which will be shown on the screen for the remaining duration of the session.

Attention sound A short attention sound will be played 500 ms before the start of every target sentence (1000 Hz tone over 200 ms).

Playback delays The distractor stories always start 2 s before the target sentence and end approximately 1s after the target speaker has finished.

Audio settings The *target sentences* have to be normalized to -22.0 dB RMS, while the *distractor stories* are set to -25.0 dB RMS. At the beginning of the game, the *target sentences* have to be presented at a SPL of 62 dB, while the *distractor stories* are played at 55 dB SPL.

Progress tracking To retrieve useful data from the training sessions, the average SRT values of each session (excluding the practice rounds) and the gained reward stickers have to be tracked.

3.4 Speech stimuli

To further maintain comparability and to avoid the additional workload of re-evaluating appropriate speech stimuli, the new application will feature the same word databases as used in the original project. Cameron and Dillon dedicated a whole section to the considerations that went into the subject of speech capabilities of children at a certain age. Only the distractor stories will have to be changed since the original ones aren't publicly available.

To mitigate the problem of neither having a native English speaker nor extensive recording equipment and expertise available, the speech stimuli will be generated by a neural Text To Speech (TTS) tool. The freely available *Voicemaker* [27] service will be used for this purpose. Alongside offering natural sounding, well recorded audio files, this approach also makes it significantly easier to add new speech stimuli into the project later on.

3.5 Visuals

Apart from the move to VR peripherals and the new audio assets, the visuals will be the biggest deviation from the original application. Since the 2D presentation of the LiSN software wouldn't be sensible here, a 3D environment has to be added.

Considering that the target participants of this project are children, the visual presentation is of great importance. If the game isn't engaging, there is a higher risk of participants not complying with the required training schedule due to the repetitive nature of the sessions[5]. The novelty of VR in general might already be helpful in mitigating this problem, but

[5] This issue was already brought up by Cameron and Dillon in the conclusion of [23]

it should still be considered during development. Additionally, VR applications require special care when it comes to orientation and sense of space within the 3D environment, otherwise there is a risk of players feeling disorientated.

Due to the limited resources during development, the use of 3rd-party assets will most likely be inevitable. To still be able to freely distribute the project, the license requirements of all non-proprietary assets have to be considered.

Matching visuals to audio Since the original audio stimuli did not include reflections, reverberations, or other sound propagation effects, they will be absent in the new application as well. This adds some restrictions to the design of the 3D environment. A big empty room would suggest the occurrence of reverberation and echo and should therefore be avoided. Ideally, an outdoor location should be selected, where free-field sound propagation would be natural. Another important point would be the visual representation of the speakers. As mentioned in *Fundamentals*, matching visual cues to the virtual placement of the sound sources, helps mitigate the inaccuracies in localization caused by the non-individualized HRTFs.

4 Concept

To reduce the risk of developing components that are incompatible with each other or are missing features, a general concept of the whole project will be created in advance. This includes an overview of all major components, their functions and how they interact with one another. Additionally, the parts that require high levels of adjustability will receive special attention.

4.1 Training Game

The actual training game is expected to feature the majority of components and assets that will be affected by alterations. Ranging from the addition of free movement to new speech and noise stimuli, there is a very broad variety of options. However, it won't be possible to consider all future additions and changes, let alone include these aspects during implementation. The risk of being forced to re-write or change larger parts of the source code to implement missing features can be minimized though by designing small and independent components with little dependencies.

Word databases As already mentioned, the *Training Game* will require the user to recognize words from randomly generated target sentences. In the current scope, only one of the lists from the original application will be included in the new program. However, the implementation of this component should be flexible enough to support different word databases without any changes to the source code. This is particularly important when considering the option of translating the application to other languages. To elaborate on how this can be achieved, *List 1* of LiSN & Learn[6] will be used as an example. Table 1

[6] All *Word Lists* of the original project can be found in Appendix A of the initial publication [2]

shows all entries from this list, with the highlighted words being the options, which have to be identified by the user in the game.

	Subject	Verb	Count	Adjective	Objects
The	**baby**	bought	**two**	big	**apples**
	boy	carried	**three**	blue	**bottles**
	clown	cleaned	**four**	broken	**cars**
	doctor	drew	**five**	green	**chairs**
	girl	dropped	**six**	little	**crayons**
	lady	had	**seven**	old	**cups**
	man	liked	**eight**	orange	**shoes**
	nurse	saw	**nine**	red	**spoons**
	teacher	watched	**ten**	yellow	**trucks**

Table 1 LiSN - List 1

When observing this table three parameters can be determined:

- Sentences length: **6 words**
- Possibilities for each group: **9 options**
- 'Selectable groups': **Subject**, **Count**, **Objects**

Of course, this could be extended further, by adding support for sentences of variable length or moving from 'selectable groups' to individual 'selectable words'. However, offering too many options might introduce new problems. Taking the previous example, supporting sentences of variable length, would affect the comparability between training sessions, since the target stimuli would deviate stronger from each other.

All three of the before mentioned parameters have to be considered when creating a dynamic framework, which is supposed to be able to support different word lists. Additionally, this example also provides an interesting anomaly. All *options* inside a given *group* are sorted alphabetically, except for the entries of the **Count** - group. It is important to not rely on any feature, like the order of the entries within a *group*, that depends on the actual content of a given database.

4.2 Asset management

Moving on to the topic of asset management, the same level of adjustability has to be maintained on the addition or replacement of assets, to properly make use of the advantages of flexible components inside the application.

Unity offers multiple ways to handle this issue. The straight forward approach would be to assign the individual asset files through drag and drop using the inspector. Even though this would in principle fulfill the requirement of changing/adding assets, it's not

ideal when many assets have to be changed at once (e.g. manually replacing all files for the word database of List 1 would be tedious).

The easiest way to replace a large number of assets is most likely to change the contents of the project folders using the operating system's file management (e.g. Windows Explorer). Additional information, like which word groups shall be 'selectable' must still be provided. This can either be done through naming conventions for the individual assets, manual input in the inspector or by using a separate file.

4.3 Progress tracking

In order to be able to monitor the progress over the course of the training duration, an option to track training data has to be added. Similar to the original project, this will include the average SRT of each session as well as the number of rewards that have been gained (see section 3.3). Apart from this, the implementation should again allow for additions or replacements of the metrics to be tracked. To avoid problems with data evaluation that can occur through the changes to the format in different versions of the application, all entries have to identifiable in the actual database through tags or similar means.

User System It has to be considered that more than one person might use the same system to perform training sessions. To make sure that the progress can be tracked individually, a user system will be established. This requires that, apart from the actual training data, credentials (username and password) have to be stored. Furthermore, access to the *Training Game* has to be restricted via a login screen, to ensure that no progress data is lost or attributed to the wrong user. Finally, the addition of new accounts should be possible from inside the application, with the restriction that each user has to be uniquely identifiable (e.g. through the prohibition of duplicate usernames).

Progress export Apart from tracking the progress data for each user, it must also be made accessible to be evaluated in external tools. For this purpose, all user data will be stored in a *.json* file. The JSON format allows to hold multiple types of data, from strings for the username to arrays of floats for the SRT values, and is easily readable as well as widely used in many different applications.

In-app visualization The application must give any user an option to keep track of their progress. This will be realized through a separate screen that features a simple graph plotting the SRT values of the number of training sessions done as well as additional space to show e.g. the current average SRT over all sessions.

4.4 Application overview

Based on the given requirements and established conceptual ideas, a basic description of the application can be created. On a general level, the project can be divided into four major functional blocks:

1. User Login & Creation
2. Main Menu
3. Training Game
4. Progress Screen

Any of these parts can be considered as an individual screen or state, that may be left on a given set of conditions (e.g. leaving the progress screen by hitting the *ESC* key). This is visualized through a simplified flowchart in figure 2:

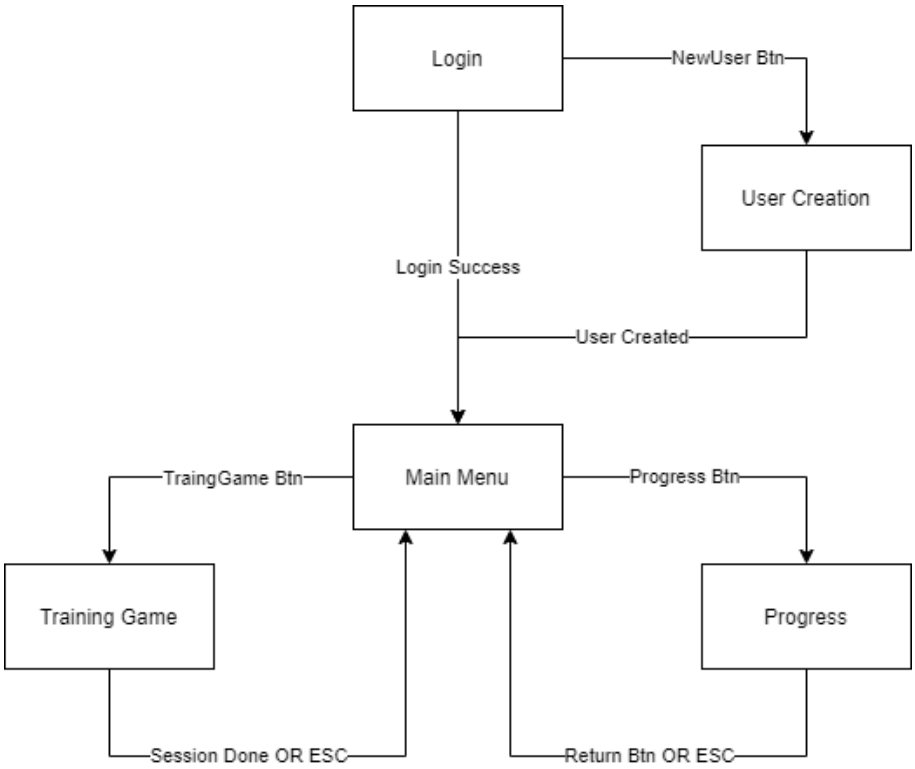
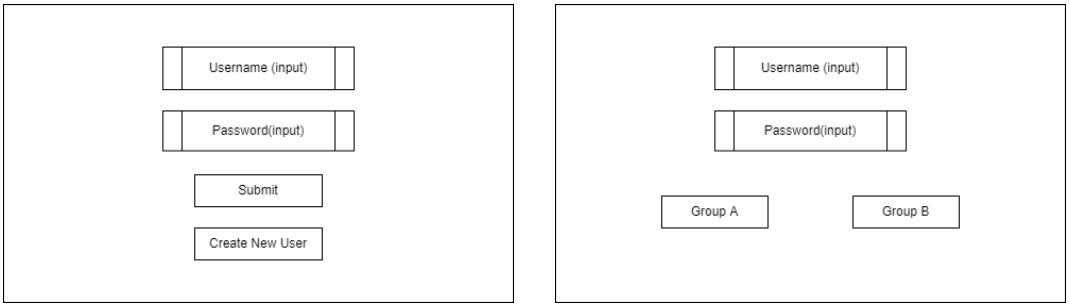


Figure 2 Overview & Transitions

Following this brief overview, the contents of each functional block will be discussed in greater detail.

User Login & Creation This will be the first screen to be rendered at the start of the application. The user has to either enter his credentials or create a new account to access any other part of the application. Figure 3 shows an outline of how the UI for both login or user creation could look like.



(a) Login (b) Creation

Figure 3 Login & User Creation

Main Menu The *Main Menu* provides access to either the *Training Game* or the *Progress Screen* and allows the user to leave the application.

Figure 4 Main Menu

Training Game Since the *Training Game* has already been discussed in section 4.1, only the UI elements of the scene will be addressed here. These consist of the already mentioned buttons for the **4 word options** and **unsure**, which are shown after the end of each sentence. Instead of displaying these elements on a separate screen, they will be rendered inside the 3D environment.

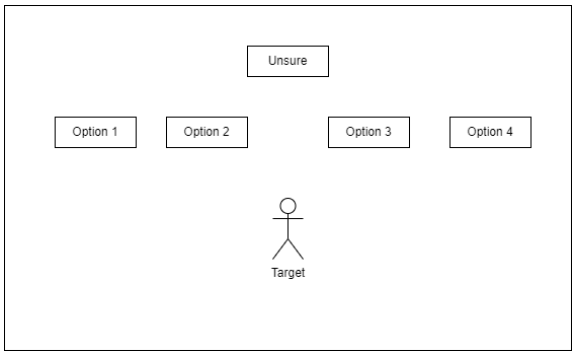


Figure 5 Training Game UI

Progress The progress screen will feature a graphical representation of the SRT values achieved by the user and the number of sessions done. Additionally, other data like the

current average SRT or the total number of sessions done can be featured as well.

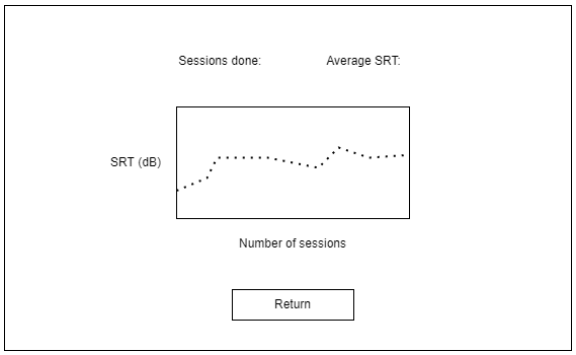


Figure 6 Progress

5 Implementation

The implementation will be documented by describing all major building blocks of the project with regard to their functionality. This also includes brief discussions of design decisions during the development process and some general explanations on how underlying systems work. At some points, a general understanding of object-orientated programming and Unity's unique functionalities (like component assignments through the inspector) will be useful to follow the reasoning behind certain approaches.

5.1 Project Overview

An empty Unity 3D example project has been used as starting point. Using this foundation, multiple scenes have been added to separate the different parts of the project, matching the description given in Concept.

- **Preload**: The first scene to be loaded. Responsible for holding general behaviors, global data and establishing fixed starting conditions.
- **Login**: Restricting access to the *Main Menu*, *Progress Screen* and *Training Game* by either requiring a successful login attempt or the creation of a new user account.
- **Main Menu**: A simple hub, from which either the *Training Game* or the *Progress Screen* can be reached.
- **Training Game**: Loading the 3D environment as well as all other required assets for the training process and running the game loop.

5.2 Preload

The concept of a *Preload* scene is rather common in Unity projects. The idea behind this, is to implement an additional scene that can be used to run initialization routines or hold general behaviors. Through the project configuration it is possible to manually set the build index of any scene in the project. Assigning the lowest possible value (0) to *Preload* guarantees that this will be the first scene to be loaded upon startup.

Usually, if a new scene is loaded, all objects from the previous scene are destroyed. This can be avoided by calling *DontDestroyOnLoad*. Any object that is marked through this method will be kept throughout the whole runtime of the application, if not destroyed manually. Using this functionality allows to establish general behaviors that are valid in every scene of the project (e.g. pressing *ESC* to access the *Main Menu*). However, special care has to be taken, if a scene can be loaded and unloaded multiple times, because this would lead to the creation of duplicated instances of the objects marked with *DontDestroyOnLoad*, if not handled correctly.

User Management *UserManagement* is the only script that can directly access and alter the contents of the user system's database. This includes functionalities like checking passwords on login attempts, adding a new account after a successful user creation, or storing training data at the end of a session. Since these features have to be accessed from a variety of components in different scenes, *UserManagement* is marked as *DontDestroyOnLoad* and carries a public static reference to itself. This allows to call methods of the object instance in *Preload* by using the class definition. Even though, *Preload* will only be loaded once at the start of the application, *UserManagement* still makes sure that there is always only one instance of this type in existence by checking the self reference.

User Data To manage the information for each individual user, the *userData* class has been created. Apart from storing required data like *username*, *password* and *training progress*, this class also offers some simple utility functions that allow to retrieve, change or add attributes of a *userData* object without making them *public*. All user accounts are stored as a *List* of *userData* objects (*userList*) that is kept in *UserManagement*.

JSON Export In order to make the information in *UserManagement* persistent and exportable, the aforementioned *userList* will be saved as a JSON file (*userData.json*). This format is not only widely used and supported by other applications, but is also directly human-readable.

Since Unity's inclusion of JSON isn't very advanced, the *JSON .NET FOR Unity* plugin is used here. This allows amongst other advantages the serialization of C# lists of custom types. The contents, which shall be included in the JSON, can simply be determined by marking an attribute as *JsonProperty*. The resulting file looks like this:

```
[
  {
    "username": "emptyUser",
    "password": "test",
    "group": 1,
    "gamesPlayed": 0,
    "rewardsGained": 0,
    "currentSNR": 0,
    "gamesSNR": [],
```

```
    "gamesRewards": []
  }
]
```

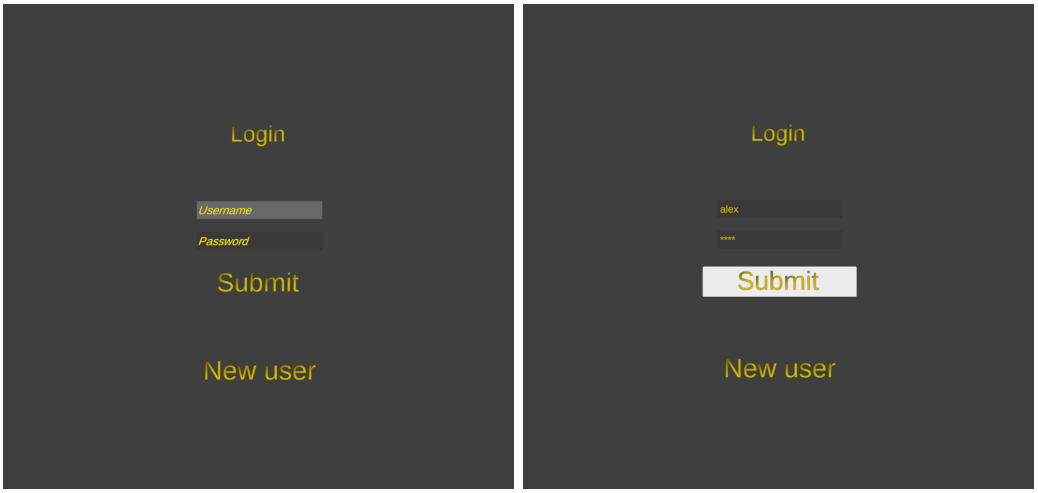
The *userData.json* file will be stored at Unity's application specific persistent data directory. At startup *UserManagement* will try to load the database from this path. If there's no file available a new database will be created and stored at the same location.

Scene Management To ensure that at the start of the application, no unwanted scene is loaded, a simple component has been added to *Preload*. *SceneManagement* iterates through all scenes in the build configuration and unloads them if necessary. An exception is made for *Login*, since this scene will be loaded directly after initialization.

5.3 Login

Login presents a simple User Interface (UI) that allows the user to either enter their credentials or create a new account. Since these functionalities might be used in multiple contexts (e.g. granting access to different parts of an application), the resulting actions after login or user creation attempts have been decoupled from the components. This has been achieved through the usage of delegates and a simple parent script. In the current use case, both the successful creation of a new user or a valid login attempt lead to the *Main Menu* being loaded.

Since the keyboard offers a simple and efficient way to enter information into text fields and typing is definitively harder when wearing a Head Mounted Display (HMD), this scene is rendered with a regular (non-VR) camera and input system. The user can either navigate the UI by selecting the elements with the mouse or switching between them by hitting the *TAB* key (as often implemented in similar use-cases). Figure 7 demonstrates how highlighting is used to communicate to the user, which element is currently selected.



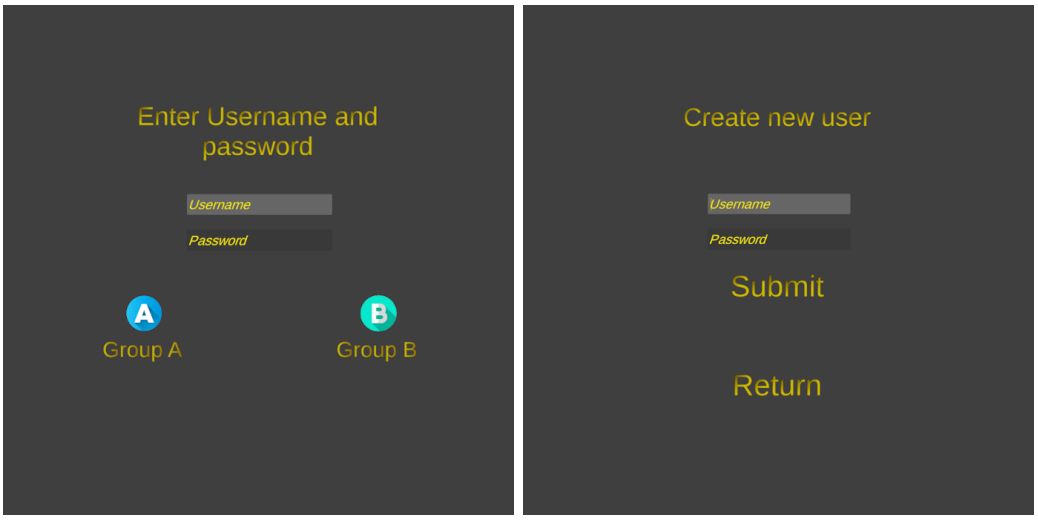
(a) Username (b) Submit

Figure 7 Login UI

When the credentials have been entered, they can be submitted by either pressing the

Submit button or simply hitting 'return', while the *password* field is selected.

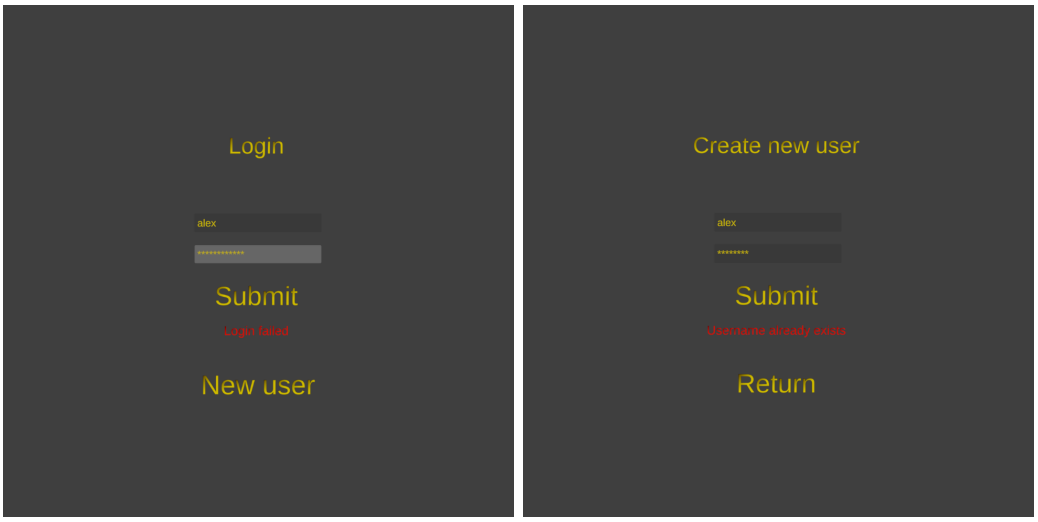
In previous iterations, the user creation also featured a group selector. This allowed to implement a control group with non-spatialized audio stimuli or separate between the usage of female or male voices. This feature has been cut from the UI, since this use-case does not exist in the original LiSN & Learn program. However, it is still included in the associated components, which makes it very easy to implement the concept of different groups again.



(a) Old (b) New

Figure 8 User creation UI

Finally, it is also important to communicate failed login or creation attempts to the user. This can be the case if a wrong username or password was entered when logging in or trying to create a new user with a name that is already in use.



(a) Login - Failed attempt (b) User Creation - Invalid username

Figure 9 Login / user creation - Error feedback

5.4 Main Menu

The *Main Menu* is the first screen that is rendered using a VR camera. This does not only affect the visuals but also the input system and, therefore, the way UI elements are selected. Instead of using mouse and keyboard, a cursor is drawn at the point where the user is currently looking. If this cursor hits any UI element, it will be highlighted and can be selected by pressing *button one* of the Oculus controller[7]. Figure 10 shows how the user can select different menu items by tilting his head and therefore moving the eye-cursor.

(a) No selection (b) Progress - Highlight (c) Training Game - Highlight

Figure 10 Main Menu - UI navigation

Since this input method includes moving the camera, the *Main Menu* is rendered within a closed 3D room (see figure 11). Otherwise, users would see 'empty space' when turning their heads too far in any given direction. This approach has been taken from the UI example scene of the Oculus Software Development Kit (SDK). This concept also allows to show multiple menu screens at once by taking advantage of the different walls of the room, which might come in handy for future additions to the *Main Menu* scene.

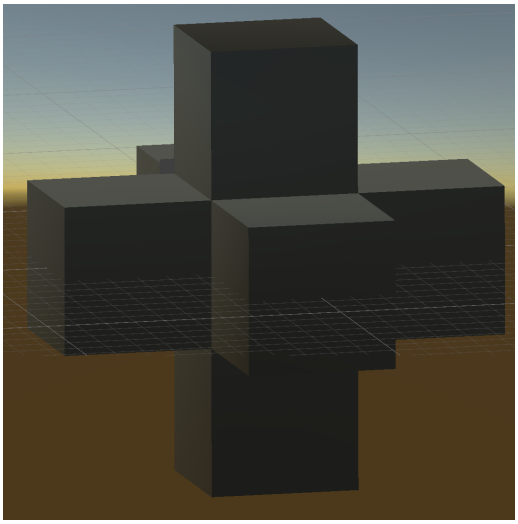


Figure 11 Room

[7] This can easily be changed to any other button, if e.g. a different VR headset is used

UI prefabs To maintain a consistent look throughout all UI elements the *prefab* functionality of Unity has been used. This allows the creation of templates that can be used to maintain the same layout and color-scheme on every occasion. Additionally, this feature also allows to easily change a certain parameter, like the font color, for all objects that are based on the *prefab*.

Progress Screen The *Progress Screen* is designed offer a simple insight into the training data of the current user. This includes some information in text form, like the total number of games played, the current average SNR, and the number of rewards gained throughout the training process. It also provides a graphical representation of the SRT values over the sessions that have been done.

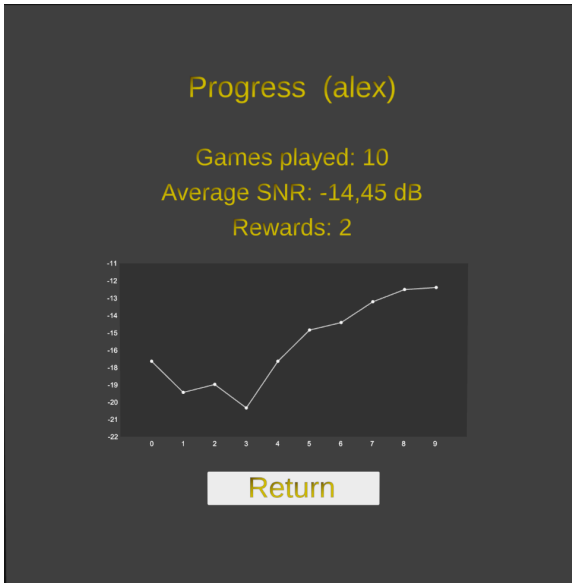


Figure 12 Progress Screen

5.5 Training Game

Before the major components are discussed in greater detail, a brief overview of the functionalities of the new *Training Game* implementation will be given. The main task stayed the same as in the original, with the user being challenged by correctly identifying single words from a target sentence against background noise consisting of two distracting stories. While the distractors are located on the left and right side of the listener, the target audio is played from a source directly facing the player. After each sentence, the user has to choose from four possible options or alternatively select *unsure*. The training sessions ends after 40 rounds have been played.

5.5.1 Training Game Manager

Operating as a master script for the *Training Game*, this component offers callback interfaces to react to different events in the game and is the only script that is allowed to trigger methods in other objects. For this to work, references to all other major components are included in the script. The callback interface is realized through the usage of delegates,

which are set in the *Training Game Manager*. This approach establishes a fixed hierarchy in the scene that makes it easier to add new components, since only one existing script has to be altered to implement the new behavior.

Parameters To easily realize smaller changes to the *Training Game*, the manager script offers a broad variety of parameters through the usage of the *SerializeField* functionality. This allows to change values of different parameters through the Unity Inspector without having to alter the source code or make the variables *public*. Figure 13 shows all adjustable entries that are accessible through the inspector.

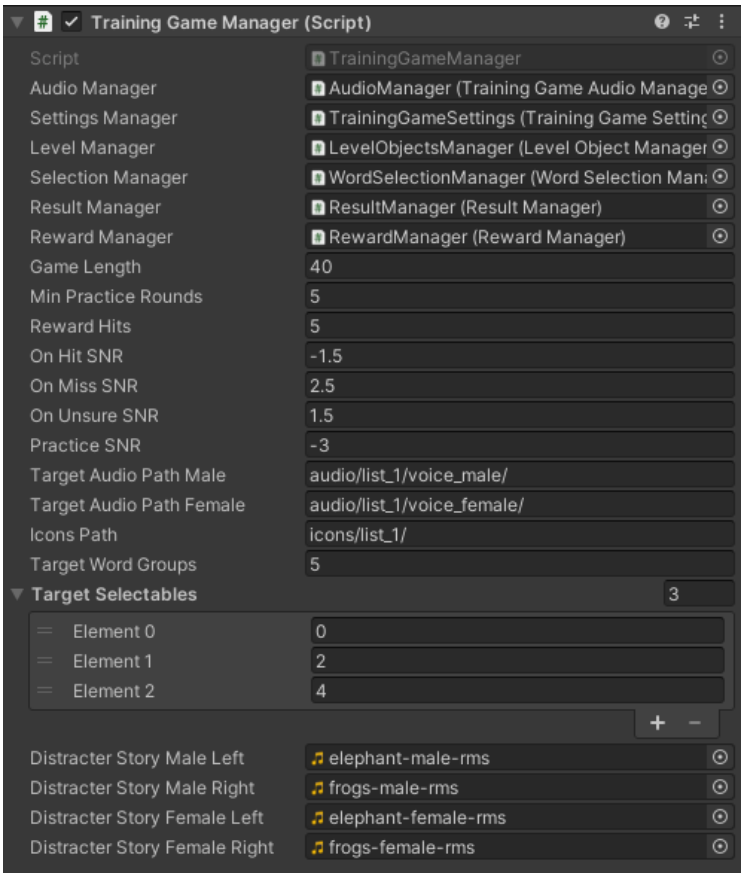


Figure 13 Training Game Manager parameters

In the following, the different callback methods, offer by the *Training Game Manager*, to react to events in the game loop will be described.

Start Derived from *MonoBehaviour*, this method gets called on the first frame after the script has been enabled. In this case, that happens right after the *Training Game* scene has been loaded. *Start* is used for setting all delegates of the other components and triggering the placement of the talker and distractor objects.

OnStart Called by *Training Game Settings* after the voice selection has been done. Create the *LiSN database* object and initialize the *AudioManager* according to the selected voice option. Afterwards, generate the first target sentence and trigger the first game round.

OnPlayingDone Triggered by the *Audio Manager* at the end of each game round. This method is responsible for fetching the required strings and icons for the player's task (guessing the correct word from a selection of four options) and hand them over to the *Word Selection Manager*. If *Unsure* was selected in the previous round, the same contents are shown again.

OnHit Play the *OnHit* sound effect, increase the variables for total *hits* and *rewardCounter* and change the SNR by *OnHitSNR* (*-1.5 dB*). If more than *rewardHits* (*5*) consecutive correct answers have been given, a rewards sticker is added to the screen and the *OnReward* sound is played.

OnMiss Play the *OnMiss* sound effect, increase the *miss* variable, set *rewardCounter* to *0* and change the SNR by *OnMissSNR* (*+2.5 dB*).

OnUnsure Play the *OnUnsure* sound effect, toggle the *repeatSentence* flag and change the SNR by *OnUnsureSNR* (*+1.5 dB*).

OnContinue Hide the *WordSelection* UI and check whether the number of rounds to be played (*40*) has been reached. If this is the case, call *OnSessionDone*. Otherwise, create a new target sentence, increase the counter variable for the number of rounds played, and start playing the next sentence.

OnSessionDone After the last round has been played, the average SRT is calculated and the *Results* screen is shown, to give an insight about the performance of this session.

5.5.2 LiSN database

This class is used to load all necessary assets for the *Training Game* using the *Resource* functionality of Unity. A single instance of this type is created and kept in the *Training Game Manager* and is responsible for holding all audio clips, icons and strings that can be used in the game. Additionally, the *LiSN database* offers some utilities to make access to the required assets easier.

The paths[8] to the desired audio and image files, as well as other data specific to the *Training Game*'s assets have to be offered in the constructor. This allows to change between different asset groups by only adjusting the creation of the *LiSN database* object. It is not necessary to alter any script in order to change the assets, since all required parameters are accessible through the inspector.

5.5.3 Audio Manager

The *Audio Manager* is responsible for starting and stopping playback of the target and distractor audio, as well as any sound effect. For this purpose, the script holds a reference to all audio sources of the scene:

[8] The files have be located in a *Resource* folder inside the applications project. The path can be simply entered as string (e.g. *audio/list1/*)

- Player (used for sound effects)
- Target
- Distractor (left)
- Distractor (right)

After the audio for a given round has been played, the *onPlayingDone* callback of the *Training Game Manager* is called. The length of a round consists of the playback duration of the current target sentences combined with the delays at the beginning and end (see *Playback delays* in 3.3). Instead of polling the state of the target audio source to catch the end of the playback and adding the given delays, a co-routine is started that waits for the required amount of time before stopping the distractors and calling *OnPlayingDone*.

Audio clip assembly Since the target sentences consist of randomly assembled audio files for each individual word, the *Audio Manager* has been provided with a method that allows to combine an array of clips into a single clip. The current implementation requires that the sampling rates of any file is the same, because otherwise the playback speed will be either too fast or too slow.

Adaptive SNR In order to easily alter the SNR after each round, a custom audio mixer for the *Training Game* has been added. It features separate channels for the target audio and both distractors, as well as a master channel and an additional player channel that is used for the playback of sound effects. The SNR is adjusted by changing the attenuation of the talker channel, while keeping a constant setting on the distractor audio. The control of the audio mixer via script has been implemented through the usage of the *Exposed Parameter* functionality.

Figure 14 Audio Mixer

5.5.4 Word Selection Manager

As mentioned before, at the end of each round the player has to select the correct word - which was part of the last target sentence - from four options. All entries will be randomly assigned to the UI elements, to ensure that the game can't be tricked by finding the correct answer through the recognition of a pattern. Once a selection has been made, the *Word Selection Manager* will either call the *OnHit*, *OnMiss* or *OnUnsure* callback. After a small delay, the previous UI will be replaced with a single button that will start the next round by triggering the *OnContinue* method.

Figure 15 shows a typical example of a selection in the game. All UI elements feature both a written and a graphical representation of their content, which improves overall readability and appearance of the interface.

(a) Options (b) Continue

Figure 15 Word Selection

5.5.5 Level Objects Manager

The *Level Objects Manager* takes the position of the *Player Camera* and places all other objects, required to run the *Training Game*, relative to it. This includes the target and distractor audio sources, with their visual representation as well as the UI elements that is used to show the word selections in each round. Through this system it is ensured that all objects are placed correctly around the player, which is especially important since the audio spatialization is dependent on the position of the objects. Another advantage is the simple option to relocate the game within either a given environment or the transition to a completely different level. Since the position of the aforementioned objects is solely dependent on the *Player Camera*, there is no need to manually adjust their location when moving the *Training Game* to a different place. However, the *Level Objects Manager* does not check for collisions with the environment, therefore it is necessary the validate if all other objects are placed correctly after altering the coordinates of the *Player Camera*.

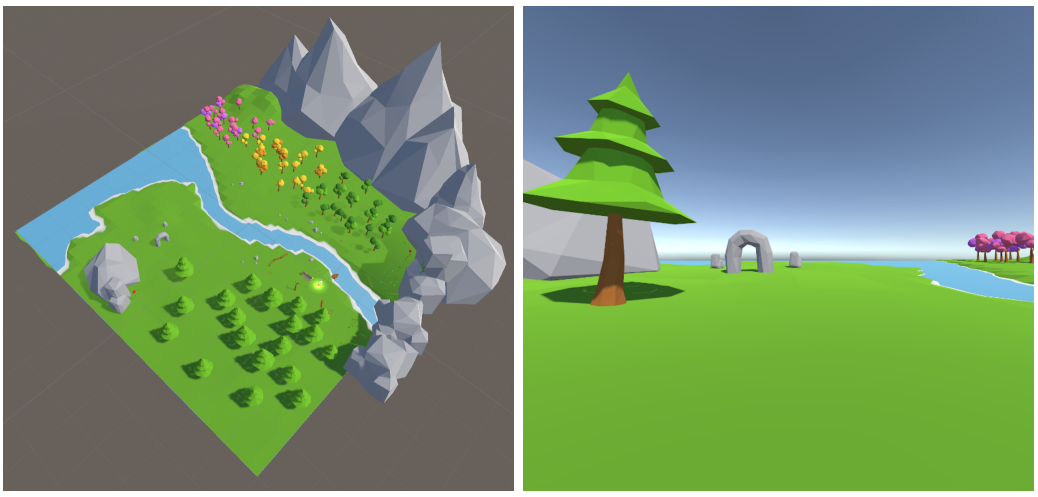
The relative positions of the talkers or the UI towards the player can be changed by

adjusting their associated 3D vectors, which are accessible through the inspector. Since the audio sources have been configured without distance based volume roll-off, it is not necessary to consider this when altering their positions.

5.5.6 Visuals

As mentioned before, the move to VR requires the addition of 3D graphics to the *Training Game*. This consists of both an outside environment and avatars for the audio sources in the scene. In both cases, free 3rd party assets[9] have been used. Since currently, neither the level nor the avatars fulfill any purpose apart from visualizing the scene, they can simply be replaced without considering any possible dependencies or similar problems.

Environment The *Simple Low Poly Nature* pack has been chosen as scenery for the *Training Game*. Mainly because the friendly looking, open-space environment fits the requirements of this application very well. The simplistic visual design and the small scale of the level are also beneficial, since they don't require a lot of computing power to be rendered and an extensive, highly detailed level wouldn't add any advantages to this use-case.

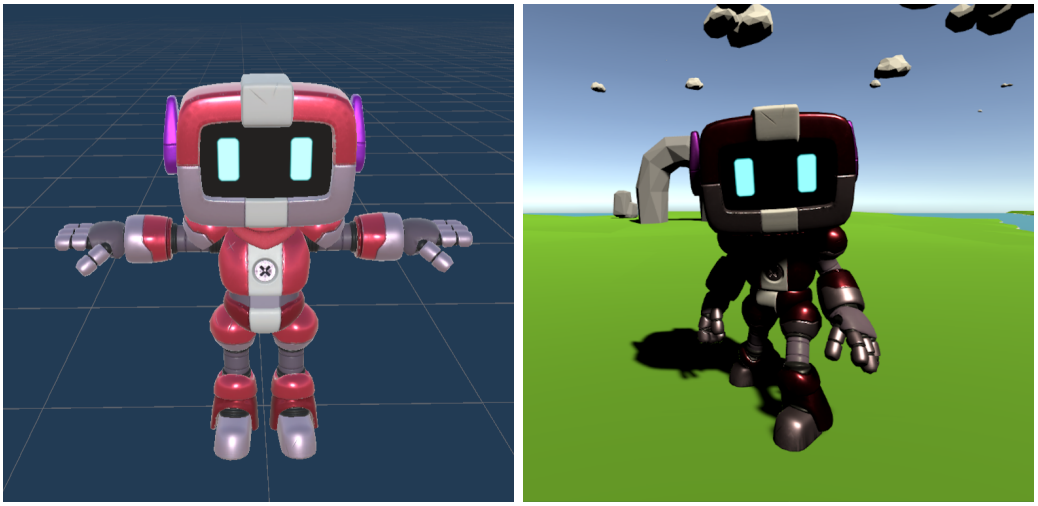


(a) Topview (b) Scene-View

Figure 16 Environment

Avatar All talkers are represented by the *Jammo Robot* model. To make them visually distinguishable, each position gets assigned a different color. The abstract design of the face helps to mitigate the lack of speech animations, while still remaining a humanoid look that is plausible for the representation of a speaker. Even though the character model comes with a small selection of animations, none of them were really fitting for the intended use-case. This problem has been found on the majority of free-to-use options for character models. However, since the avatar fulfills no purpose other than visually representing the talkers, it can be easily replace if a better option is found.

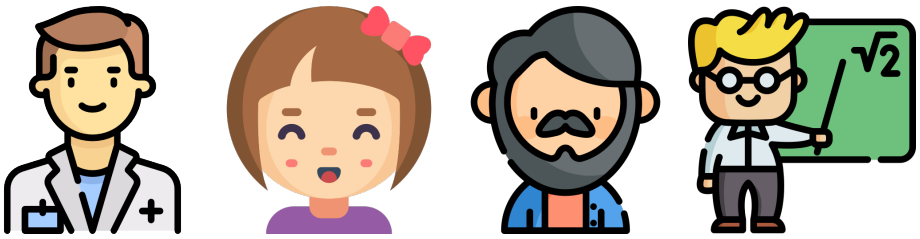
[9] All 3rd party assets that are used in this project are mentioned with their respective sources in Appendix.



(a) Prefab (b) In Scene

Figure 17 Avatar

Pictograms The pictograms that are used in the *Training Game* have been selected to clearly represent their associated context and fit the overall look of the application. Ideally any UI element that also includes an icon should be understandable without the addition of any text. Figure 18 shows a small selection of pictograms that had been chosen to represent words of the *Subject* group (see section 4.1).



(a) Doctor (b) Girl (c) Man (d) Teacher

Figure 18 Pictogram examples

5.6 Audio Editing

As already described, all speech stimuli are generated using the free online TTS synthesizer *Voicemaker*. The female recordings were generated using the *Kendra* voice, while *Joey* was used for the male recordings. In both cases, .mp3 was chosen as audio format with a sampling rate of 24 kHz[10]. To achieve more fluent results, the stimuli have been generated as whole sentences. In total nine sentences from "The baby bought two big apples" to "The teacher watched ten yellow trucks" have been created for both voices.

For the distractor stories *The Frogs Who Wished for a King* and *The Thief and the Elephant*[11] had been chosen. Since *Voicemaker* has restrictions towards the length of text

[10] The option to use higher sampling rates (44.1 kHz or 48 kHz) is restricted to premium access and therefore was not selected here.
[11] The links to the full texts are referenced in the Appendix

that can be translated at once, the stories had to be converted to speech in blocks and then be combined into a single audio file.

Afterwards, both the target and distractor stimuli had been normalized using the *Root Mean Square (RMS) Normalize* plugin for Audacity[12]. As described in section 3.3, distractor stories are set to -25 dB RMS, while target sentences have to be set to -22 dB.

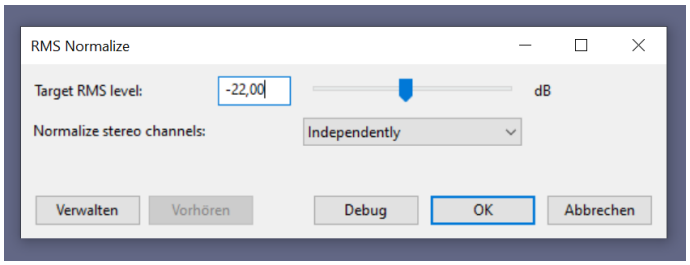
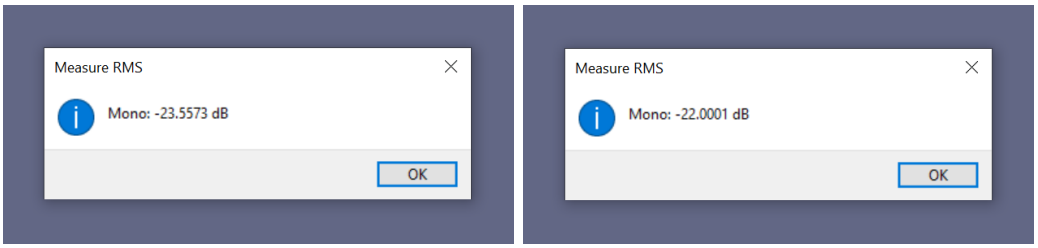


Figure 19 RMS normalization

In order to verify, if the normalization has worked as intended, the audio clips have been analyzed using Audicity's *Measure RMS* tool. Figure 20 shows a before and after comparison for an example clip.



(a) Pre (b) Post

Figure 20 RMS measurements

In the next step, the individual words had to be cut from the full sentence audio clips. This was accomplished by creating a separate track for each word while keeping the original file in the same project unaltered. Through this approach, it was possible to evaluate whether the cuts had been set correctly, by listening to the single words back against the full sentence repeatedly.

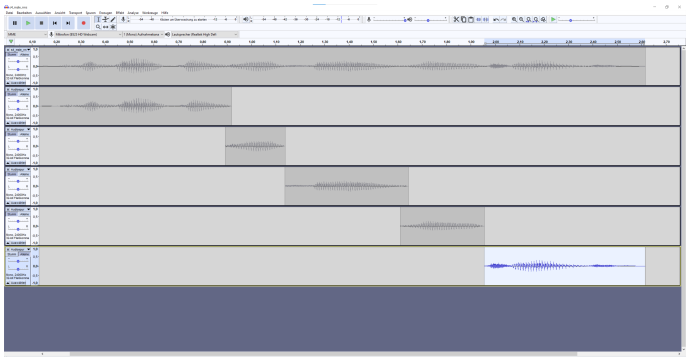


Figure 21 Target word editing

[12] See table 7 in Appendix

In several cases, there was no pause between the end of one and the beginning of the next word. To avoid hard cuts at the edges of these clips, linear fading has been added. The extent to which this was done, has been based on manually adjusting the degree of fading until the clips sounded sufficiently good.

6 Discussion

The greatest concern regarding the efficacy of the new training program lies in the usage of the Oculus spatializer. Even though the developers advertise realistic 3D audio rendering, the implementation has been designed with a focus on gaming applications, which typically don't require very precise auditory localization. However, through the inclusion of visual representation for all speakers and the wide spatial spread between the different sound sources, small inaccuracies will probably be compensated for.

If these measures are proven to be insufficient to address this problem, or if the underlying project shall be used for a different test or training program that comes with more strict requirements towards precise audio spatialization there are some additional options that should be considered. One possibility would be the replacement of the Oculus spatializer with another audio renderer that offers higher accuracy. Apart from this, the inclusion of individualized HRTFs would also allow for improved localization results, with the drawback of significantly higher expenses since they require elaborate measurements for each user.

Another aspect to be considered lies in the usage of TTS for the generation of speech stimuli. Even though neural TTS offers strongly improved results compared to regular speech synthesis, professionally made individual recordings would still provide a more natural sound and, therefore, a more realistic challenge. Since the replacement of audio assets in the project is rather simple, this issue could be investigated through a small test study.

A calibration process is not necessary for this training program, due to the adaptive nature of the procedure and the usage of the same type of stimuli for both target and distraction. For other use-cases this will most likely not be the case. Depending on the requirements of the test or training application either a calibration routine or at least the usage of fixed hardware should be considered.

7 Conclusion

Over the course of this project, the original LiSN & Learn auditory training software has been analyzed, described and a concept for a new iteration was designed. Furthermore, a brief introduction into the subject matter and some underlying principles has been given. Following the approach of creating an extensible, open-source application that remains comparable to the original, while featuring a 3D environment and support of VR peripherals, a new implementation of the training program has been developed.

Apart from recreating the existing software, this project was also used to establish a foundations to validate to which extend the development tools provided by Unity and Oculus are suitable to realize auditory test and training programs. This target is also reflected in the design principles that have been addressed in Approach. The available data for LiSN & Learn can be used as a reference point, to validate the efficacy of the new framework and the extensible design of the application is advantageous when realizing different test or training scenarios on the established platform.

Outlook Several aspects that have been mentioned throughout the report already show a lot of potential for future work. The most important task is the conduction of a performance evaluation of the new application. Based upon the results either the mitigation of issues in the existing application should be addressed (e.g. by improving the audio rendering) or the implementation of other test or training programs could be started.

Once this topic has been concluded, the inclusion of more advanced training scenarios can be evaluated. The established framework would allow for way more complex listening scenarios that could in principle simulate an actual cocktail party, with a large amount of speakers and different noises either in a free-field location or a closed room. Another interesting aspect lies within the introduction of unique feature of VR applications, like the built-in head tracking options. This could allow to add dynamic cues to localization tasks and therefore take another step towards establishing more realistic test and training scenarios.

Abbreviations

API	Application Programming Interface
ASHA	American Speech-Language-Hearing Association
CAPD	Central Auditory Processing Disorder
DSP	Digital Signal Processing
HMD	Head Mounted Display
HRTF	Head Related Transfer Function
IDE	Integrated Development Environment
ILD	Interaural Level Difference
ITD	Interaural Time Difference
JSON	JavaScript Object Notation
LiSN	Listening in Spatialized Noise
LiSN-S	Listening in Spatialized Noise - Sentences Test
MAMA	Minimum Audible Movement Angle
RMS	Root Mean Square
SDK	Software Development Kit
SNR	Signal to Noise Ratio
SPD	Spatial Processing Disorder
SPL	Sound Pressure Level
SRT	Speech Reception Threshold
TTS	Text To Speech
UI	User Interface
VAE	Virtual Acoustic Environments
VR	Virtual Reality

List of Figures

List of Tables

References

[1] American Speech-Language-Hearing Association (ASHA). *Central Auditory Processing Disorder*. 2021. URL: https://www.asha.org/Practice-Portal/Clinical-Topics/Central-Auditory-Processing-Disorder/.

[2] Sharon Cameron and Harvey Dillon. "Development and Evaluation of the LiSN & Learn Auditory Training Software for Deficit-Specific Remediation of Binaural Processing Deficits in Children: Preliminary Findings". In: *Journal of the American Academy of Audiology* 22 (2011), pp. 678–696. DOI: 10.3766/jaaa.22.10.6.

[3] Sound Storm CAPD Pty Ltd. *Soundstorm Webpage*. 2021. URL: https://www.soundstorm.app/.

[4] Barbara G. Shinn-Cunningham. "Spatial hearing advantages in everyday environments". In: *Proc of ONR workshop on Attention, Perception, and Modeling for Complex Displays, Troy, NY*. 2003.

[5] Tanya L. Arbogast, Christine R. Mason, and Gerald Kidd. "The effect of spatial separation on informational masking of speech in normal-hearing and hearing-impaired listeners". In: *The Journal of the Acoustical Society of America* 117.4 (2005), pp. 2169–2180. DOI: 10.1121/1.1861598.

[6] Gary Jones and Ruth Litovsky. "A cocktail party model of spatial release from masking by both noise and speech interferers". In: *The Journal of the Acoustical Society of America* 130 (2011), pp. 1463–1474. DOI: 10.1121/1.3613928.

[7] Adelbert W. Bronkhorst. "The cocktail-party problem revisited: early processing and selection of multi-talker speech". In: *Attention, perception & psychophysics* 77 (2015), pp. 1465–1487. DOI: 10.3758/s13414-015-0882-9.

[8] Piers Dawes and Dorothy Bishop. "Auditory processing disorder in relation to developmental disorders of language, communication and attention: A review and critique Research Report". In: *International journal of language & communication disorders / Royal College of Speech & Language Therapists* 44 (2009), pp. 440–465. DOI: 10.1080/13682820902929073.

[9] Richard Tyler et al. "Initial Development of a Spatially Separated Speech-in-Noise and Localization Training Program". In: *Journal of the American Academy of Audiology* 21 (2010), pp. 390–403. DOI: 10.3766/jaaa.21.6.4.

[10] Joseph Psotka. "Immersive training systems: Virtual reality and education and training". In: *Instructional Science* 23 (1995), pp. 405–431. DOI: 10.1007/BF00896880.

[11] Sam Kavanagh et al. "A systematic review of Virtual Reality in education". In: *Themes in Science and Technology Education* 10.2 (2017), pp. 85–119.

[12] Huriye Atilgan et al. "Integration of Visual Information in Auditory Cortex Promotes Auditory Scene Analysis through Multisensory Binding". In: *Neuron* 97 (2018), pp. 640–655. DOI: 10.1016/j.neuron.2017.12.034.

[13] Frederic L. Wightman and Doris J. Kistler. "Resolution of front–back ambiguity in spatial hearing by listener and source movement". In: *The Journal of the Acoustical Society of America* 105.5 (1999), pp. 2841–2853. DOI: 10.1121/1.426899.

[14] Jens Blauert. *Spatial Hearing: The Psychophysics of Human Sound Localization*. The MIT Press, 1996. ISBN: 9780262268684. DOI: 10.7551/mitpress/6391.001.0001.

[15] G. Christoph Stecker and Frederic J. Gallun. "Binaural hearing, sound localization, and spatial hearing". In: *Translational perspectives in auditory neuroscience: Normal aspects of hearing* 383 (2012), pp. 383–433.

[16] Bill Gardner and Keith Martin. *HRTF Measurements of a KEMAR Dummy-Head Microphone*. 2021. URL: https://sound.media.mit.edu/resources/KEMAR.html.

[17] Rob Drullman and Adelbert Bronkhorst. "Multichannel speech intelligibility and talker recognition using monaural, binaural, and three-dimensional auditory presentation". In: *The Journal of the Acoustical Society of America* 107 (2000), pp. 2224–2235. DOI: 10.1121/1.428503.

[18] Christopher Berger et al. "Generic HRTFs May be Good Enough in Virtual Reality. Improving Source Localization through Cross-Modal Plasticity". In: *Frontiers in Neuroscience* 12 (2018), p. 21. DOI: 10.3389/fnins.2018.00021.

[19] Mirza Beig et al. "An Introduction to Spatial Sound Rendering in Virtual Environments and Games". In: *The Computer Games Journal* 8 (2019), pp. 199–214. DOI: 10.1007/s40869-019-00086-0.

[20] Jack Scott. "Effects of auditory training on hearing aid acclimatization". In: *The Journal of the Acoustical Society of America* 120 (2006). DOI: 10.1121/1.4781399.

[21] Georgia Cambridge et al. "Auditory training for adults with cochlear implants: A systematic review". In: *International Journal of Audiology* (2022), pp. 1–9. DOI: 10.1080/14992027.2021.2014075.

[22] Ruth Litovsky. "Spatial Release from Masking". In: *Acoustics Today* 8 (2012), pp. 18–25. DOI: 10.1121/1.4729575.

[23] Sharon Cameron, Helen Glyde, and Harvey Dillon. "Efficacy of the LiSN & Learn auditory training software: Randomized blinded control study". In: *Audiology Research* 2 (2012), pp. 86–93. DOI: 10.4081/audiores.2012.e15.

[24] Judith Pokorni, Colleen Worthington, and Patricia Jamison. "Phonological Awareness Intervention: Comparison of Fast ForWord, Earobics, and LiPS". In: *Journal of Educational Research - J EDUC RES* 97 (2004), pp. 147–158. DOI: 10.3200/JOER.97.3.147-158.

[25] Unity Technologies. *Unity Plans and pricing*. 2021. URL: https://store.unity.com/#plans-individual.

[26] Facebook Technologies LLC. *Oculus Audio Spatializer Features*. 2022. URL: https://developer.oculus.com/documentation/unity/audio-spatializer-features/.

[27] Voicemaker. *Webpage*. 2021. URL: https://voicemaker.in/.

Appendix

Pictograms

Baby	https://www.flaticon.com/free-icon/baby_822123
Boy	https://www.flaticon.com/free-icon/boy_163834
Clown	https://www.flaticon.com/free-icon/clown_879970
Doctor	https://www.flaticon.com/free-icon/doctor_921059
Girl	https://www.flaticon.com/premium-icon/girl_2632445
Lady	https://www.flaticon.com/free-icon/woman_921023
Man	https://www.flaticon.com/premium-icon/man_2423809
Nurse	https://www.flaticon.com/free-icon/nurse_2637720
Teacher	https://www.flaticon.com/free-icon/teacher_2436654

Table 2 Pictograms: List 1 - subjects

Two	https://www.flaticon.com/free-icon/2_4020060
Three	https://www.flaticon.com/free-icon/3_4020062
Four	https://www.flaticon.com/free-icon/4_4020064
Five	https://www.flaticon.com/free-icon/5_4020076
Six	https://www.flaticon.com/free-icon/6_4020077
Seven	https://www.flaticon.com/free-icon/7_4020078
Eight	https://www.flaticon.com/free-icon/8_4020079
Nine	https://www.flaticon.com/free-icon/9_4020080
Ten	https://www.flaticon.com/free-icon/10_4020082

Table 3 Pictograms: List 1 - numbers

Apples	https://www.flaticon.com/premium-icon/apple_3137044
Bottles	https://www.flaticon.com/premium-icon/wine-bottles_3437507
Cars	https://www.flaticon.com/free-icon/car_2555013
Chairs	https://www.flaticon.com/free-icon/dining_2271494
Crayons	https://www.flaticon.com/free-icon/crayons_1685441
Cups	https://www.flaticon.com/premium-icon/cup_1166261
Shoes	https://www.flaticon.com/free-icon/sneakers_1785348
Spoons	https://www.flaticon.com/premium-icon/spoon_2447668
Trucks	https://www.flaticon.com/premium-icon/delivery-truck_2107330

Table 4 Pictograms: List 1 - objects

Conversation	https://www.flaticon.com/free-icon/conversation_942751
Trophy	https://www.flaticon.com/free-icon/trophy_3112946
Parrot	https://www.flaticon.com/free-icon/parrot_1025368
Cat	https://www.flaticon.com/premium-icon/cat_2173478
A	https://www.flaticon.com/premium-icon/letter-a_3665909
B	https://www.flaticon.com/free-icon/b_3600910
Next	https://www.flaticon.com/free-icon/next_190520

Table 5 Pictograms: others

Sound effects

DM-CGS-45 (Hit)	https://assetstore.unity.com/packages/audio/sound-fx/free-casual-game-sfx-pack-54116
Gameshow wrong answer fail buzzer beep (Miss)	https://www.dreamstime.com/stock-sound-fx-gameshow-wrong-answer-fail-buzzer-bleep-gameshow-wrong-answer-fail-buzzer-bleep-sound-effect-audio128203414
Game Win 08 (Reward)	https://www.dreamstime.com/royalty-free-music-game-win-sound-effect-game-development-game-win-audio119138215
sci-fi-beepbutton-119-sound-effect-19119263 (Unsure)	http://freesoundeffect.net/sound/sci-fi-beepbutton-119-sound-effect

Table 6 Sound effects

Other assets

JSON .NET For Unity	https://assetstore.unity.com/packages/tools/input-management/json-net-for-unity-11347
Oculus Intergration	https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022
Audacity RMS normalization plugin	https://forum.audacityteam.org/viewtopic.php?t=95421
The Thief and the Elephant	https://www.studentuk.com/2020/05/13/the-thief-and-the-elephant/
The Frogs Who Wished for a King	https://www.studentuk.com/2016/07/20/the-frogs-who-wished-for-a-king/

Table 7 Other assets

3D assets

Simple Low Poly Nature Pack	https://assetstore.unity.com/packages/3d/environments/landscapes/simple-low-poly-nature-pack-157552
Jammo Character	https://assetstore.unity.com/packages/3d/characters/jammo-character-mix-and-jam-158456

Table 8 3D assets