HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

FACHSTELLE ASN
ALKOHOL- UND DROGENPRÄVENTION
IM STRASSENVERKEHR

AM STEUER
NIE

Virtual Buzz:
Simulating visual Influences of Alcohol in an Augmented-Reality App

Term Project

Department of Computer Science
University of Applied Science Rapperswil

Fall Term 2016

Authors:	Konrad Höpli, Roberto Cuervo-Alvarez
Advisor:	Prof. Oliver Augenstein
Project Partner:	Fachstelle Am Steuer Nie (ASN), Zürich

Table of Contents

1 Abstract

1.1 Introduction

The industrial partner ASN (which roughly translates to "never [drink] behind the wheel”) provides information regarding the influence of alcohol and other drugs on humans. To make adolescents aware of the impact of alcohol on the human brain, they use a variety of tools such as (drunk) driving simulators and glasses, which come with impairments, to provide the visual experience of being drunk.

Due to the desire to appeal to their young target audience, ASN sees a lot of value in using leading edge technologies to communicate their message. In case of this project, the goal was to experiment with Google Cardboards and to investigate their current capabilities with regards to being able to simulate some of the visual disturbances caused by alcohol.

1.2 Approach

As a first step, we decided on using the Vuforia augmented reality SDK for the game engine Unity. This way, we had the option to use the additional features coming with both Unity and the SDK made therefore. Especially the various image effects and filters provided by Unity looked very promising for our intentions.

The second step was developing a first prototype of an AR app, which simply used the ‘blur’ component provided by Unity on the running smartphone camera and thereby proofed the plausibility of the project idea. We then went on to attempt implementing the other visual effects described by ASN. Based on the research nature of this project, the whole development process as well as the information gathered on the topic of both Unity and AR capabilities is supposed to be documented well. In a manner to not just give a good insight on the project itself, but the topic and used tools as well. It is aimed at both potential future contributors to this project as well as developers interested in testing the water of augmented reality with the tools used here.

1.3 Results

This project resulted in a functional app which can be used by the attendees of ASN presentations to simulate the visual experience of being drunk. It serves as a proof of concept for the initially uncertain idea to simulate the visual effects caused by alcohol in AR.

The app also identified the currently still severe limitations of the processing power of smartphones regarding the used technologies.

The documentation serves as introduction to the world of AR and Unity while also giving insight on the project itself, the encountered limitations and possible extensions or future projects this could lead to.

2 Introduction

This chapter covers an overview of the project as well as the underlying idea and the goals and objectives contained therein.

2.1 Overview

This project was set out to be primarily for research purposes regarding the available technologies and features regarding the uprising "Google Cardboard"-device since neither of the participating parties had any previous experiences with this new area of development. The company "Am Steuer Nie" (ASN) has provided the basic idea for the project, which is to create a new mobile app replacement for the actual glasses with built-in sight impairments they use for demonstration purposes of the influences of substances like alcohol to the human sight (and driving) and possibly use those in a sort of a game to catch the attention of their target audience. The app is not (yet) planned to be released in any app-stores, since that is not essential to the project even though it could be an option for future versions so they can be used in other companies similar to ASN or simply educational purposes in general.

The documentation of our findings and development experiences regarding the new technologies is also an essential part of this project since it is supposed to provide an inexperienced developer the needed insight into the area of virtual and augmented reality as well as game development with the Unity game engine.

2.2 Goals and Objectives

The goals and objectives of this project initially were very vague and partially had to be determined further over the course of the project as we acquired a greater insight on the available tools and possibilities we had. The basic goal laid out was to simply develop some kind of app to simulate the visual impairments caused by the influence of substances such as alcohol while providing a good documentation for all the gained knowledge over this new area of programming as well as all the technologies that had to be used to achieve that.

Initial Requirements	**Fulfillment status**
Working app for simulating substances	Fulfilled
App for Android-OS	Fulfilled
Usage of "Google Cardboard"	Fulfilled
Developing concrete Game Ideas	Partially fulfilled
Extensive Documentation	Fulfilled

Table 1: Initial Requirements Table

In addition to the basic objectives, we were able to split up to whole project further as we went on and identified the following goals along the way:

Identified Goals and Objectives	**Fulfillment status**
Implementing visual effects	
Blurred View	Fulfilled
Inconstancy	Fulfilled
Tunnel View	Fulfilled
Red Light Weakness	Fulfilled
Duplications	Not fulfilled
Flash/Night Blindness	Fulfilled
Ability to configure Settings	Fulfilled
Configuration Management	Fulfilled
Understanding used Features & Shaders	Partially fulfilled
Implementing a custom image effect	Partially fulfilled
Performance Optimization	Fulfilled
Including virtual Objects	Partially fulfilled
External Controller	Not fulfilled
User Manual	Fulfilled

Table 2: Identified Goals and Objectives

3 Background

This chapter is intended to give a slight insight into the background of both the students as well as the industrial partner involved in this project.

3.1 Fachstelle Am Steuer Nie (ASN)

The industrial partner Am Steuer Nie, ASN for short, (which roughly translates to: "Never behind the wheel") is a rather small company with the goal to teach people between the ages of around 14 and 22 about the negative influences of substances like alcohol and drugs on the human capabilities especially regarding driving. The focus regarding substances thereby lies on alcohol and they have a variety of tools at their disposal in order to provide a somewhat realistic and immersive experience.

3.1.1 Motivation

A part of the motivation to develop a new AR-App simply comes from making use of the newest technologies available and expanding the tools available to continue making the simulation of the big variety of effects, which alcohol and other substances can have on our well-being, more realistic. Along with that aspect also comes the 'hype-factor' that these still relatively new kinds of technologies typically have on the younger generations and therefor possibly improves the reception of both marketing and the actual demonstrations.

3.1.2 Existing Tools

ASN currently uses a set of physical tools to provide a realistic experience and get their message about the bad combination of the influence of substances and driving across. The following tools merely serve as examples and do not reflect all of what they have at their disposal.

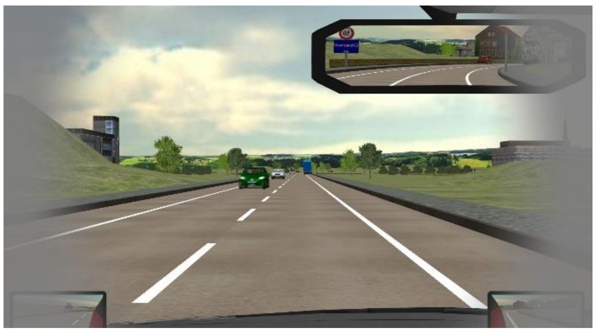


Figure 1:(Drunk) Driving Simulator Preview

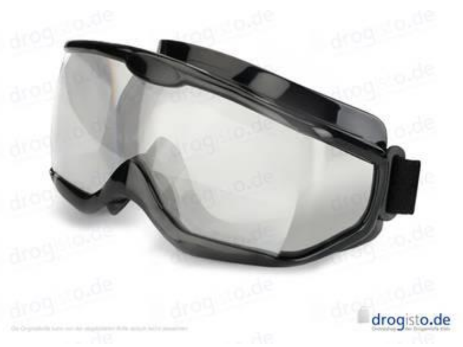


Figure 2: Alcohol Simulation Goggles [1]

3.2 Personal Motives

This chapter is supposed to give a somewhat detailed insight into why we personally chose this project for our term project without much hesitation.

3.2.1 Konrad Höpli

Aside from the opportunity to really get into some of the newest technologies on the market, I also have been enthusiastic about gaming and its connection to the real world, which really is what AR tries to tie together on a new level. I think in most games the primary goal is to either create a really immersive and fictional experience (e.g. Life is Strange, Ori and the blind Forest) or being as realistic as possible (e.g. various kinds of simulations and even war-games like Battlefield 1). In the future I strongly believe, that the first kind will move over to virtual reality and the second kind has potential to be continued in augmented reality. However, looking at the current experiences available and the first impression of the limitations still out there, I will definitely hold on to the statement, that this is still futuristically speaking.

The other part of my motivation for this project comes from my personal lifestyle that really values both healthiness and responsibility. The responsibility aspect has basically always been important to me and surely has a relation to how I was raised on the farm of my father. Although I do drink some alcohol from time to time and consider a good, cold beer one of the best drinks available, I am heavily against all kinds of excessive substance consumptions with a negative impact on our ability to, well, function. Especially if it also potentially effects not just ourselves but other living beings.

I am also very grateful for being able to take on this project together with my good friend Roberto and our advisor, Prof. Augenstein. I have had the pleasure of getting to know Roberto really well over the last couple of years and projects. During these I have not only learned from his vast experience, but also found out, that we tend to complement each other rather well. My experiences with Prof. Augenstein during most of my math courses have also been very pleasant and influenced my motivation. Even though mathematics is not one of my strong suits, his ability to explain complex algorithms in a simple (or IT related) manner have resonated well with me.

3.2.2 Roberto Cuervo-Alvarez

I can divide the roots of my motivation for this project in three big categories:

Technological reasons: These are obvious: the possibility of testing the waters of the world of virtual and augmented reality and learning its technologies could not be more interesting. Although we have seen that these technologies are not yet fully developed and product-ready during this project, the simple act of working “on the cutting-edge” of technology was very exciting.

Philanthropic reasons: I am not a drinker and I personally suffered a loss in my youth exactly because of the relationship between alcohol and cars. So, it was really appealing to me to seize the opportunity of trying to help ASN in the battle against alcohol behind the wheel. Even the smallest effort counts and if one young person remembers his/her experience with our app in a given moment and then decides not to drive while being drunk, this would be more than enough for me.

Personal reasons: My personal reasons are obvious for me too. The opportunity of working with my classmate and friend Konrad could not be better. During the weeks we worked together, I think we have become a good team and in spite of unexpected diseases, I think we made a good job. I also cannot forget Prof. Augenstein. I have known him before attending this school and visited some of his math lessons and profited greatly from his knowledge, patience and support.

4 Different Realities [2]

One of the goals of this project is the development of an app that should be used to visualize the negative visual effects that substances just like alcohol can have and this is best done by creating an experience as immersive as possible. One of the current trends regarding immersion is the usage of augmented and virtual realities with the help of some of the newest tools available such as Google Cardboard, Gear VR and the HTC Vive or Oculus Rift. The first ones depend on the processing power of modern smartphones whereas the latter ones rely on computer hardware optimized therefore. All of devices remain rather uncommon, but their popularity surely is rising [2].

The following graphic provides a quick overview about the relationship of the available options using a simplified version of the reality-virtuality continuum described in published papers as early as 1994. While those terms may already have been introduced way before that time, the impact and availability of these technologies has only been surging extremely during the latest years.

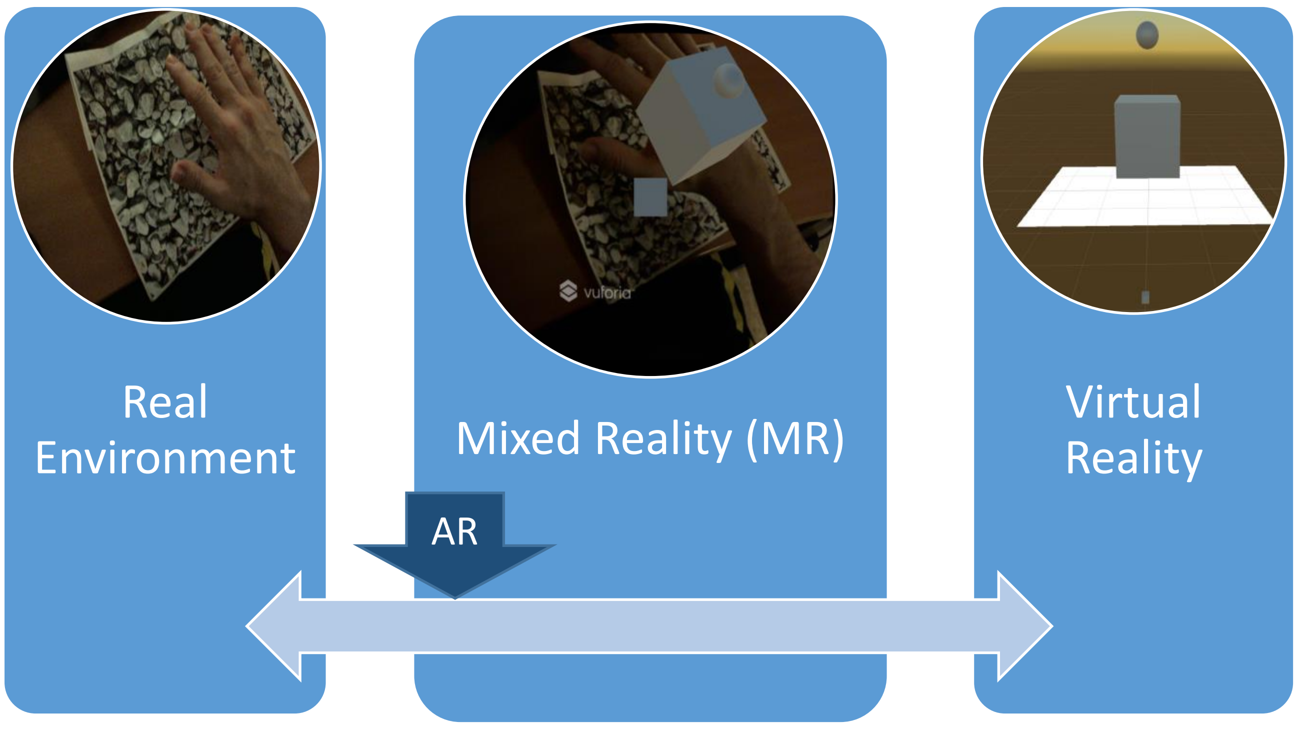


Figure 1: Simplified Reality-Virtuality Continuum

On the left side of the relationship, there is what we consider the real world as itself (or seen through a display) including existing objects and visuals, but also the constraining laws of physics that apply to them etc. On the right side is the virtual reality, which stands for a completely synthetic manifestation of a world, where its creator is in full charge of the objects existent therein as well as the governing laws they are bound to (or not).

In between the two extrema of this continuum, there is a whole range of possible implementations originally defined as mixed reality (MR), but nowadays mostly referred to as augmented reality (AR) as it is the most common. Depending on where exactly on this relationship an application or game intends to be, it will typically find itself somewhere in between with a focus on either the real world or the virtual reality. In the paper used as a source for this chapter, the points closer to the real world are defined as augmented reality whereas the points closer to the virtual reality are called augmented virtuality. The latter has still not established itself as a common term or application, but may still be to come using the newest technologies we now have at our disposal. A good example for the latter would be a completely constructed, virtual reality where a couple of real objects are integrated into and may even be interacted with.

A key factor about this formulation is, that the point on the continuum is defined by the extent to which the human perception is being altered instead of the mechanism through which the modification is achieved.

In our project’s case, we will try to primarily integrate the real environment using smartphone cameras as an optical see-through-device and image altering technologies to modify the real-world-experience captured and generate a simulation of drunkenness. As such, this project fits perfectly into the original definition of AR since we stay relatively close to the left side of the continuum.

5 Vuforia

Vuforia is a software development kit (SDK) aimed at making AR development more accessible and it provides individual versions for Android, XCode (iOS) as well as Unity.

In the beginning stages of this project we had to choose one of the versions provided and since the one for Unity comes with additional features and even though we have no previous experience with the popular game engine we quickly decided to go for that one and use the opportunity to further expand our developer horizon.

During the development of our project a new partnership between Unity Technologies and PTC Inc. was announced [3] with the goal to make AR-development more accessible by integrating the Vuforia toolset into the Unity IDE. Needless to say, this also makes it very likely that the already convenient toolset provided will continue to be improved and expanded in the future.

6 Unity

Unity is one of the most popular cross platform game development tools, also referred to as game engine, used to create video games. It consists of an IDE as well as a visual editor which can be extended by the use of plugins and components, mostly referred to as assets.

Unity itself has a quite low learning curve to get started, which is most likely one of the reasons why it became so popular recently. Other reasons are it being very well documented and free, which makes it very user friendly and attractive for users without prior programming knowledge. Besides its good documentation, it has a strong community behind it which allows new users to access the Unity forums to solve their programming issues with the assistance of more experienced programmers, or to discuss solutions and share acquired knowledge.

6.1 Basics

As already indicated, Unity is more than a simple tool and comes in separate parts, which individually contribute to both the toolset available as well as the overall user experience. It is one of the most popular game-engines available today and thanks to its licensing model especially popular among indie- as well as hobby developers. The following graph puts into perspective, how many of the most popular free mobile games in the first quarter of 2016 have been made with the game engines available:

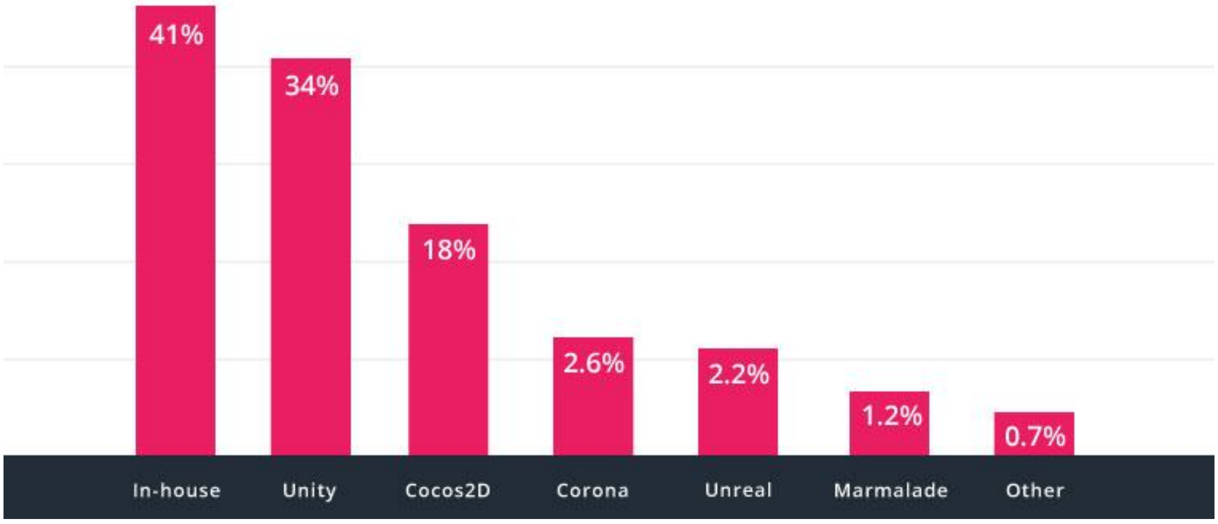


Figure 3: top 1000 free mobile games by engine [4]

While this graphic provides a decent idea of the popularity of Unity especially among free games, it does not show what the engine is actually capable of. Iin order to get a better idea of that, it is recommended to have a look at the platform [5] available to promote games made with Unity.

Game Engine

The game engine is what makes sure the created games can be run on a specific platform or environment, and in the case of Unity this is possible for basically all popular gaming-platforms such as PC, Android, Xbox as well as PlayStation.

Integrated Development Environment (IDE)

The IDE provided and developed by Unity Technologies really embraces a convenient concept that is not just for professional developers, but also easily understandable for users without a programming background even though games are generally seen as one of the most complex technical challenges available.

One of the primary reasons it does not require a development background is the separation of concerns used by Unity. The IDE itself is not intended to write a single line of code, but uses a very component based approach set in hierarchical scenes and customized using the configurable properties in the inspector window. The components are also referred to as “assets” and are one of the aspects, where the unity-developer-community really shines thanks to the integrated asset store, where anyone can share their developed assets (e.g. images, 3D models, animations, sounds and toolkits like the Vuforia SDK) with others at a price or completely for free. The amount of user collaborations is really impressive, but depending on the search term you may also struggle with finding what you are looking for since the description as well as the actual quality is of course very individual.

These components are one of the core concepts used by Unity and are really great for (but not limited to) making games due to the possibility to make each individual model used reusable as well as extendable. So, in case your game contains two humans, you are free to choose between two completely individual models, or the usage of a so called "prefab", where you assign specific properties upon each instantiation, and can add components such as behaviors to both of them as you desire. This can lead to cases where using a simple cube as an obstacle is as simple as using a car- or plane-model for the same purpose.

Code Editor

The Coding IDE that comes with Unity is called MonoDevelop and provides a rather simple, straightforward experience. It is listed as a separate part of the IDE here, since it is rather loosely coupled to the Unity IDE in that it only opens when the user wants to edit a script component and developers are actually free to choose any other coding environment such as VisualStudio.

Unity allows scripting in C# (Version 4.0), Unity Script (JavaScript for Unity) as well as Boo. Due to the low usage of Boo, the scripting manuals and documentation are as of the version 5 of Unity only focused on the other languages available.

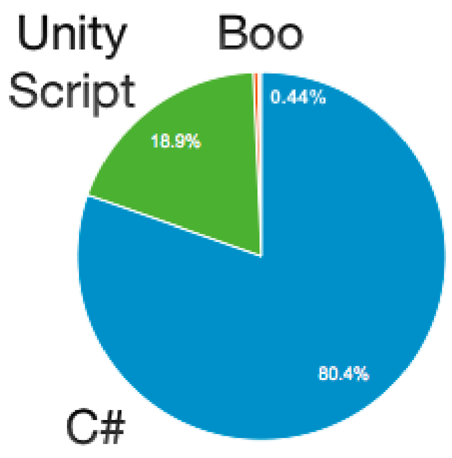


Figure 4: Distribution of Scripts created by language [6]

The individual scripts you may or may not implement throughout the development of your project will then simply be used just like any other component with a few minor tweaks. For example, any public properties in your C# script as well as the ones manually marked with the attribute "FieldSerializable" will show up in the properties window and be configurable. Individual methods defined with your script can also be assigned to the respective events such as a button-click and will then be delegated once triggered.

Our project has primarily been focused on C# scripting as well as the usage of a couple of the provided image filters and shaders (which are typically made in C Graphics or Cg in short).

7 Visual Effects

A crucial goal of this project is the implementation of a realistic simulation of what it means to be drunk or under the influence of other drugs using the technological toolset available at the time. Due to the focus on the experience and cooperation, we relied on the information provided by our industrial partner and did not go into scientific detail.

Many effects of alcohol and other substances such as balance issues, delayed reaction times and motor functions of course are important too, but these are not the focus of this project. Some of them may also be experienced when simply wearing a VR headset. This heavily depends on the individual as well as the duration during which it is worn though and is by no means calculatable.

7.1 Information about Alcohol Effects on Vision Capabilities

Our industrial Partner ASN has provided us with information on the most essential visual effects of alcohol as well as other drug related substances. This information served this project as a basic element as we tried to implement as many of those effects as possible while also trying to keep the simulation as realistic as possible. The effects listed below are ordered based on the priorities we have assigned them in cooperation with ASN.

7.1.1 Blurred View

Similar to the increase in reaction times, it takes longer to capture and focus the surroundings while intoxicated. This can result in a blurry vision especially when the eye is not trying to actively focus on a particular object.

7.1.2 Inconstancy

All of the impairments mentioned in this chapter are not behaving in a predictable manner or always at the same ‘values’, but may change over time. It also is a possibility, that everything seems fine and clear as long as one remains sitting still. As soon as movement gets involved and the number of things to process drastically increases though, the substance related influences may start to manifest.

7.1.3 Tunnel View

The human field of vision encloses usually 180 degrees. Sadly, you can only see sharp in a much smaller field of view (around 50 degrees) than that. Regardless of that, the peripherical vision is very important because it helps you react to warning colors or unexpected movements. Under alcohol influence these 180 degrees reduce steadily, being of course especially dangerous in overtake actions, spotting pedestrians, etc.

7.1.4 Red Light Weakness

The ability to differentiate between different shades of red is very limited while drunk. This is especially dangerous regarding traffic signals and rear/break lights.

7.1.5 Duplications

The human eyes both take individual images, which are then used to calculate the actual 3D image received. Under the influences of substances, this calculation can be off and result in certain items appearing to exist multiple times right next to each other.

7.1.6 Flash/Night Blindness

Human pupils change their size based on the lighting circumstances. The time needed by the pupils adapt to changes thereof is typically longer under alcoholic influence and can lead to a sort of blindness for up to multiple seconds. This is period in time that easily can lead to a crash while driving.

7.2 Image effects in Virtual Buzz

This chapter contains an explanation of the actual effects used within the app. Although the chapter naming is adjusted to match the individual components in our Unity project, the order in which they are explained matches the one used in the previous chapter (see 7.1).

7.2.1 Blur

The blurriness of the overall image provided by the camera in the app was mainly achieved through the usage of two effects:

Blur (optimized) [7] [8]**:**

This component implements a form of the "Gaussian smoothing" or "Gaussian blur" effect. This effect is rather common in image processing and primarily used to reduce the detail or noise of an image. This is achieved by selecting small groups of pixels in an image and then filtering out or smoothing the individual pixels with high contrast to the identified center to make the group as a whole appear more alike or unified. As mentioned, this can help with noise reduction so if e.g. an image had a few pixel errors (completely white) distributed over the image and this process was applied, the color of those pixels (but also others) would be changed to something more similar to their surroundings.

The overall reduction of detail within an image while still appearing somewhat close to the original is what we use this component for in our app. the option to change the following values:

- Downsample
  This could be used to heavily reduce the image quality before applying the actual effect with the goal to be more efficient while losing more detail. We decided not to use this feature, since the image captured by the camera should still closely resemble the real world as well as possible and is not of exceptional quality to begin with.
- Blur Area
  This parameter refers to the size of the pixel groups used in the smoothing process This value was the focus in our app, since it results in the most simple and efficient way to produce the desired blur effect. The user has the option to change the used value himself within the configuration.
- Blur Iterations
  The amount of iterations defines the number of passes through the actual blur process. In an effort to optimize the performance and since the adjustment of the blur area already resulted in a sufficient result, we left this value at one iteration.

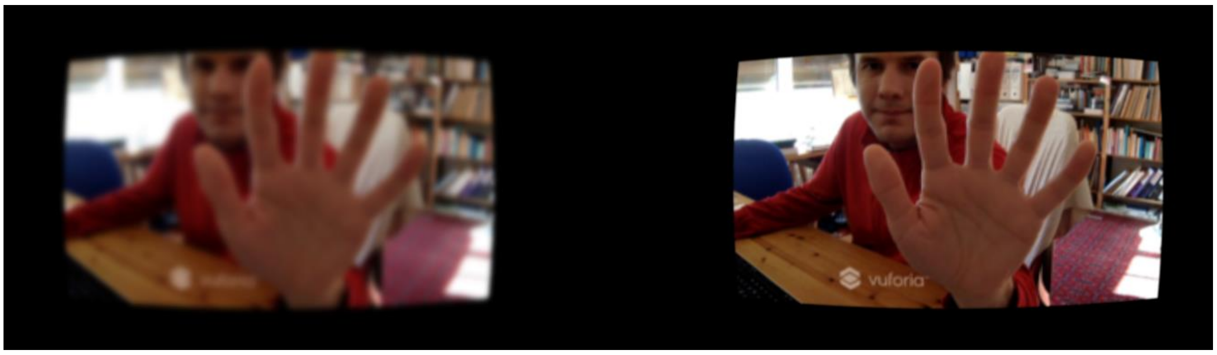


Figure 5: Blur Effect Preview

Camera Motion Blur [9] [10]**:**

As the name indicates, this component is applying a form blurriness based on the direction as well as the speed at which the camera is moving around. It is achieved by calculating a trail for individual pixels from the previously displayed images visualized in a directional blur. The goal is to provide a smoother effect for swift movements instead of the single frames well suited for games with a first-person perspective.

In our app, this effect merely acts as a supplement to the default blur effect as well as the possible side effects from using a smartphone camera. We stuck with the default values provided for all aspects except for the velocity scale. That parameter influences the velocity that has to be reached before triggering the motion blur effect and to which extent it is applied. So, since we are in an AR environment and controlling the camera with our head, we cannot expect movements comparable to the ones with a mouse controlled and increased that value.

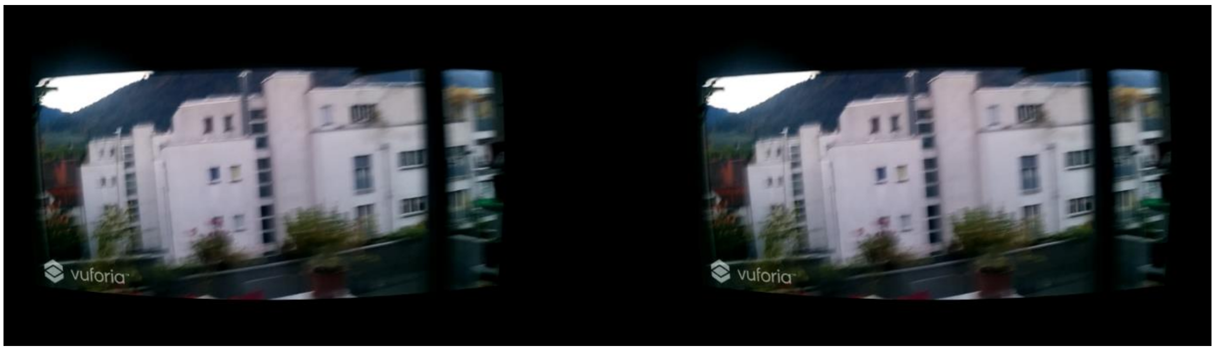


Figure 6: Motion Blur Preview

7.2.2 Randomization [11] [12]

This component was entirely built to provide the experience described in the inconstancy of individual effects while under the influence of substances. It currently applies to the blur and tiltshift components, since those are where the action happens primarily.

The algorithm used for this is called random walk and partially based on the Brownian motion for particles. It essentially performs changes in small steps based on a function rather than assigning a completely new random value to the component. In our case, the probability for each step to remain on the same value is defined as 30%. The remaining probability of 70% is then distributed among the chances of increasing and decreasing while taking into account how close to a boundary the value is. So, directly after initializing the random walk, the distribution is 50/50 leaving the chance to increase the value at the same chance of decreasing (35%). As the current value gets close to one of the boundaries of the possible range, the chance to get even closer to the boundary gets smaller and reaches 0% if the value is equal to it. With the current configuration, the changes potentially happening on each step are barely noticeable, but there is a very noticeable difference between the lower and upper end of the possible range.

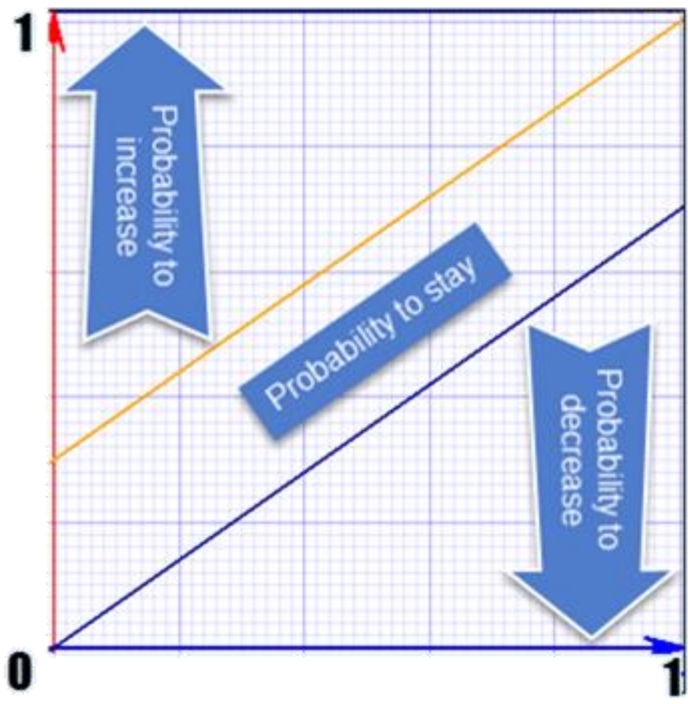


Figure 7: Graphical Visualization of the used Random Walk

7.2.3 Tilt Shift

By wearing a headset such as Google Cardboard and solely looking through the camera of your smartphone already acts as a form of tunnel view compared to our usual vision. This component is used to create a form of an additional tunnel vision by blurring out the parts of the image based on their distance to the center. This is meant to closely resemble that limited ability to focus while under the influence of substances.

The tilt shift component in Unity comes with the following options for configuration:

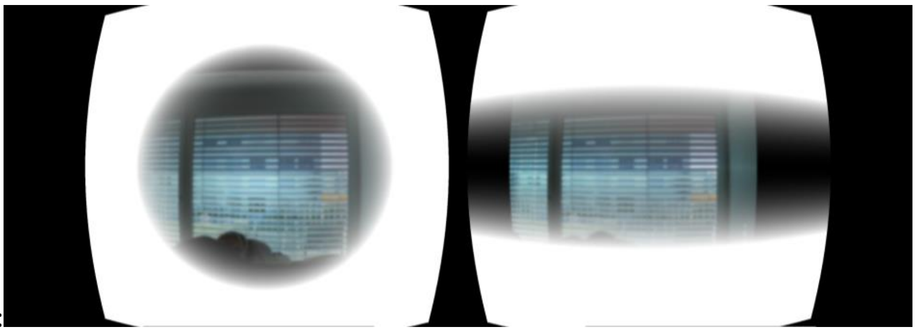
- Mode
  This setting defines whether the distance to the center of an image used for blurriness should be measured vertically or radially. So, the concrete options available are called "vertical defocus" and "radial defocus" and their calculated blur areas can be seen in the following picture:

  Figure 8: TiltShift Modes preview

  We considered both of these modes viable, but ended up using the vertical defocus, since the effect is only supplementing the already limited field of view within the AR app.

- Quality
  This is referring to the number of samples taken to calculate the individual blur effects. Because this would affect performance when using high instead of normal quality and we use the effect in combination with a regular blur anyways, we went with normal.
  The third option is simply made for preview and renders only the blurred areas as displayed in the image above.
- Max Blur Size
  The maximum blur distance is defined by this setting and we simply used the maximum of the available values in our configuration.
- Blur Area
  This refers to both size and strength of the blur applied in the designated areas. Aside from the regular blur effect, this was the second point of focus in our app and this value is therefore exposed to the user in the configuration of the app.

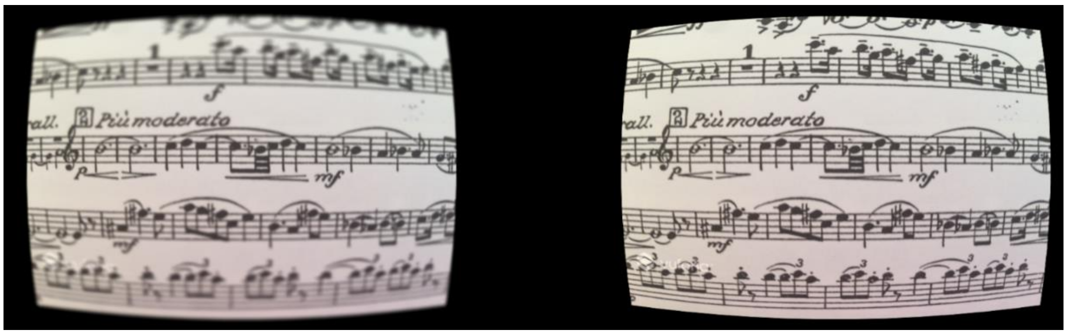


Figure 9: TiltShift Preview

7.2.4 Color Correction

This Image effect tries to simulate the weakness of differentiating between the different shades of red. In order to obtain this visual effect, we use the Color Correction Curves[1] feature from the Unity Image effects.
Color Correction Curves make color adjustments using curves for each color channel (RBG). Depth based adjustments allow you to vary the color adjustment according to a pixel's distance from the camera.
Curves work on each of the red, green and blue color channels separately and are based around the idea of mapping each input brightness level (i.e., the original brightness value of a pixel) to an output level of your choice.

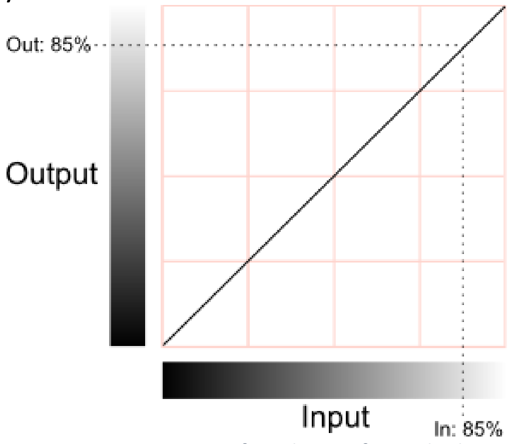


Figure 10: Mapping Options of Brightness for each Channel [13]

The horizontal axis represents the input level and the vertical represents the output level. Any point on the line specifies the output value that a given input is mapped to during processing. In our case, we choose to edit the red channel curve and configured its output in a way that the visibility of the red color is reduced and looks more like black. As you can see in the image below, it is difficult to recognize whether a traffic light is actually illuminated (the light in the image is a bike brake light).
The collateral effect of this curve is the reinforcement of the green color, but ASN told us that this "green effect" complements well the red light weakness because this weakness is stronger during the night. And the green effect slightly darkens the whole vision, so this small side effect is considered acceptable.

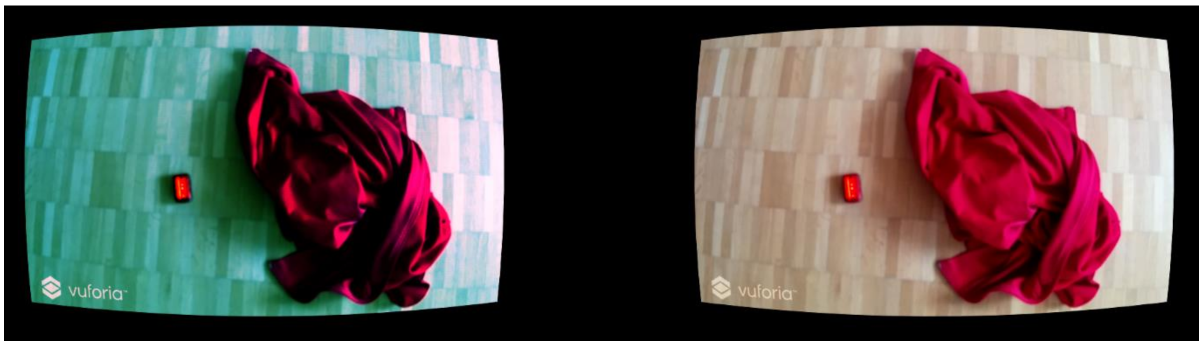


Figure 11: Preview of the used Red Color Distortion

[1] https://docs.unity3d.com/Manual/script-ColorCorrectionCurves.html

7.2.5 Delay

Depending on the smartphone used, merely using the running camera to look around already comes with a noticeable or even significant delay. This component was created by us to provide an additional delay and to partially acts as a proof of concept for our understanding of image effects in Unity.

This component is called after all the rendering and other image effects have completed. It contains a collection of frames, to which the current image is always added when it is called and enabled. Once that collection reaches a specified limit (in our case 15), the first texture of the collection is displayed on the running camera instead of the current texture. The specified limit was chosen by testing the feature on the smartphone used for this project (Nexus 5X) and creating a delay slightly bigger than a second.

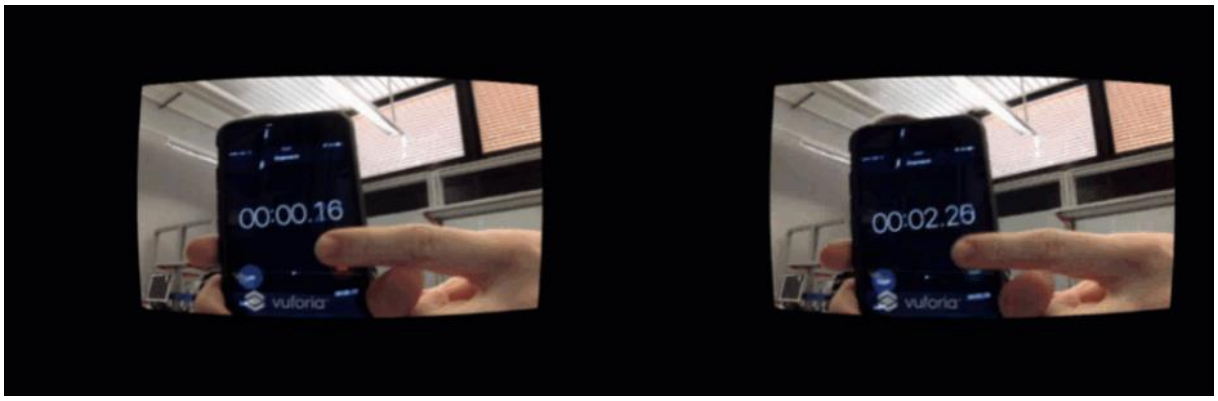


Figure 12: Delay Effect Preview

8 Solution/Implementation

The goal of this chapter is to provide a better insight into the development and structure of the Virtual Buzz app.

8.1 General Information

All the documents and projects developed over the course of this project are publicly hosted on GitHub under the following Link[2]

Project Name	Purpose
Hsr-semester-arbeit-doc	All documents and files related to the documentation of this project.
SimulatingSubstancesApp	The project of the developed app eventually named Virtual Buzz.
Hsr-semester-arbeit2016.github.io	An informal development blog written during the development to keep track of our progress and thoughts. The actual blog can be viewed on this address[3].
VrObjects	An experimental app using a combination of our first attempt at the Virtual Buzz app and some virtual objects as well as the Vuforia ImageTarget feature.
VisualEffectSelector	An experimental app, which was our first attempt at using multiple image effects at the same time and allowing them to be configured.
TiltShiftArApp	An experimental app, which was our first attempt at using the Vuforia ARCamera and the Unity TiltShift image effect.

Table 3: GitHub Project Overview

8.2 Development Environment

Within this chapter, we provide information on the software and tools used throughout our development process.

Unity Project(s):

Software	Version
Unity Engine	5.50f3 Personal
Vuforia SDK for Unity	6.1.17 Developer
Android SDK	24.4.1
Microsoft Visual Studio Enterprise 2015	14.0.25431.01 Update 3
GitKraken	1.9.3 Pro

Table 4: Unity Development Environment

Project Blog:

Software	Version
Jekyll	3.3
GitHub Pages	-

Table 5: Blog Development Environment

[2] https://github.com/HSR-Semester-Arbeit2016

[3] https://hsr-semester-arbeit2016.github.io/

8.3 Code Project (C#)

This chapter attempts to provide a good look into what the C# project of the Virtual Buzz app consists of. This is primarily achieved by using the CodeMaps generated with VisualStudio and a small description on the idea behind the created namespaces. The following image contains all of the custom C# content including the legend for the symbols used therein:

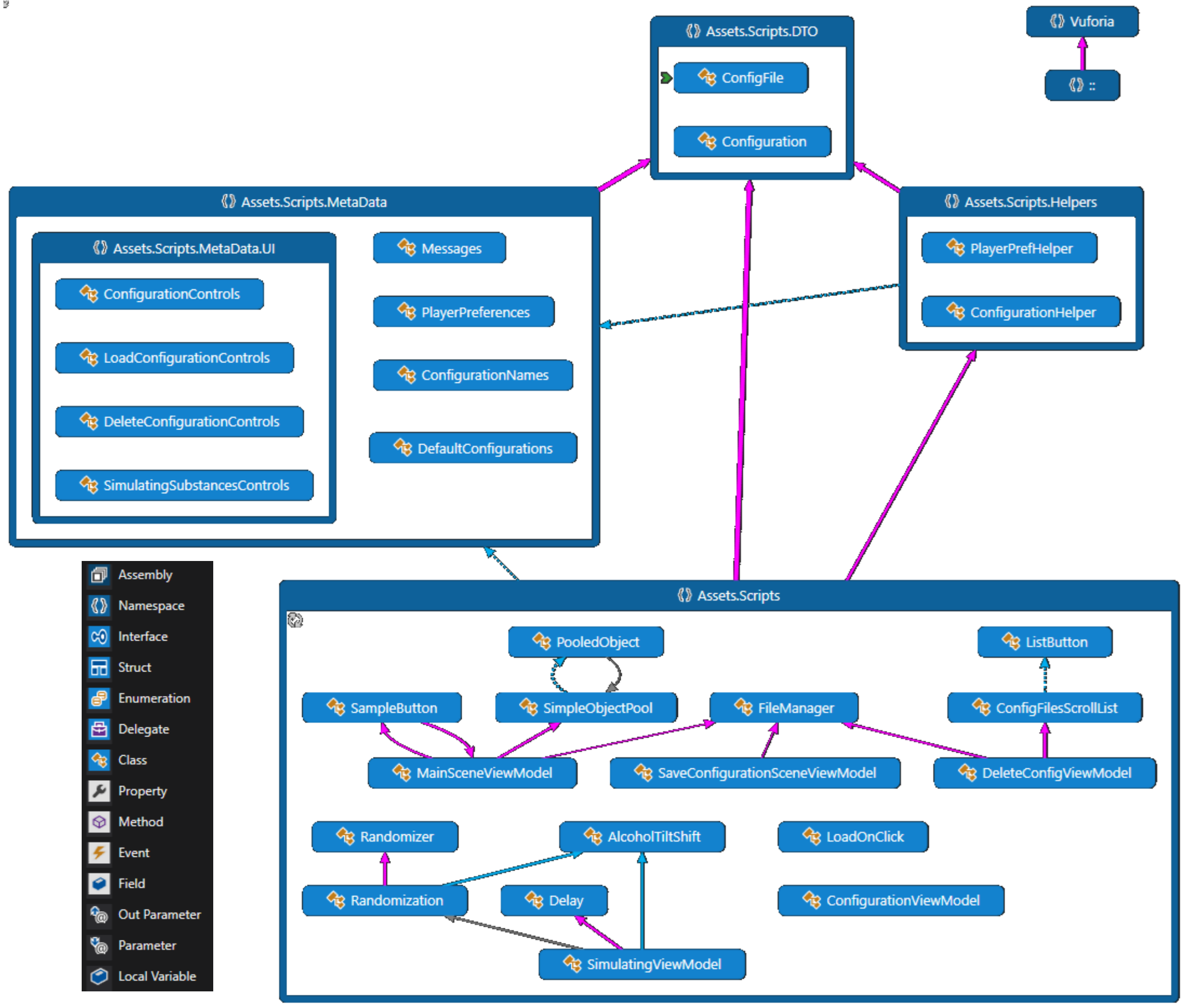
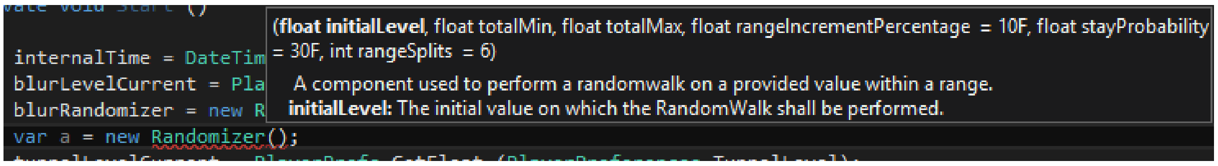


Figure 13: Code Map Overview

The more detailed part of the code documentation can be found within the code itself using the XML-comments available in C# just like this:

```
        /// <summary>
        /// A component used to perform a randomwalk on a provided value within a range.
        /// </summary>
        /// <param name="initialLevel">The initial value on which the RandomWalk shall
be performed.</param>
        /// <param name="totalMin">The absolute minimum that may be reached (usually
given by the underlying component of the initial value).</param>
```

Aside from the ability to read this information provided by scanning through the classes, this kind of comment can also help the user use the components by providing the information while coding in a suitable IDE such as VisualStudio:



```
internalTime = DateTim
blurLevelCurrent = Pla
blurRandomizer = new R
var a = new Randomizer();
```

(**float initialLevel**, float totalMin, float totalMax, float rangeIncrementPercentage = 10F, float stayProbability = 30F, int rangeSplits = 6)

A component used to perform a randomwalk on a provided value within a range.
initialLevel: The initial value on which the RandomWalk shall be performed.

Figure 14: VisualStudio XML-Comment Preview

8.3.1 Scripts

This folder contains all the custom scripts built for and used by our app and thus most of the functionality. All classes with the suffix "ViewModel" are primarily interacting with the GUI-elements of the scenes within the app. Due to the various kinds of classes and their different complexities, there is no further general information provided here, but code as well as the comments within will offer a more detailed insight.

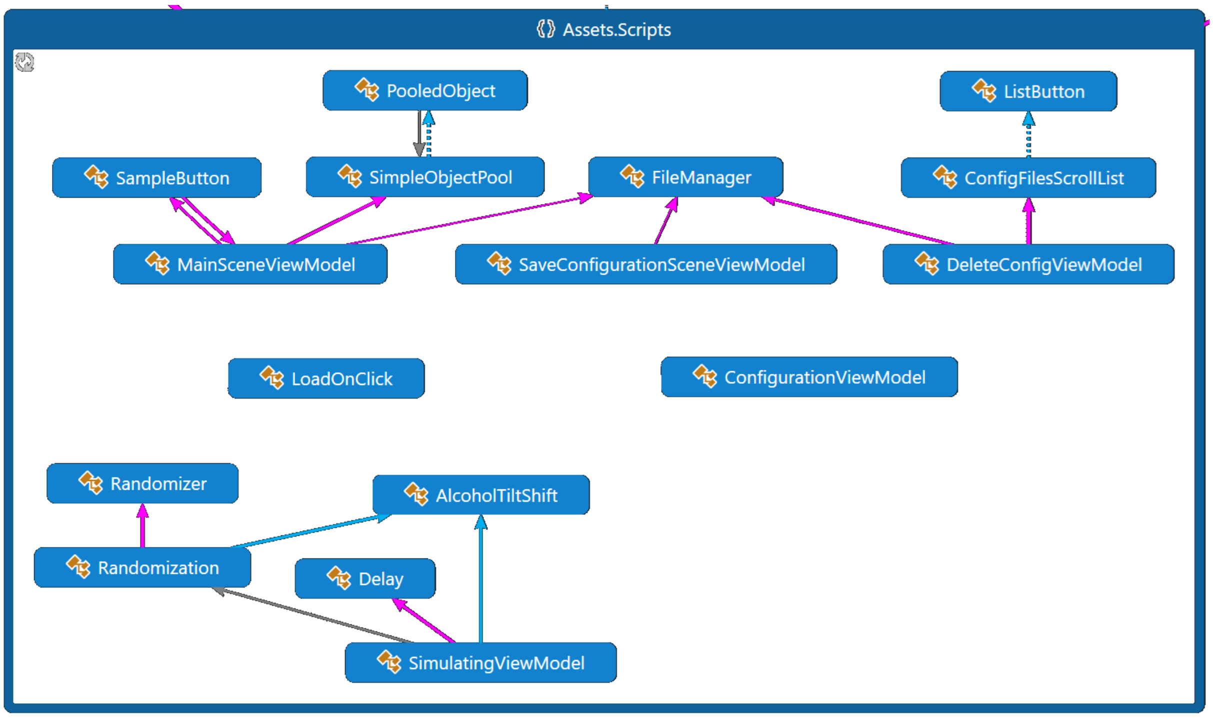


Figure 15: CodeMap Scripts

8.3.2 DTO

This namespace consists solely of the two data transfer objects used for configurations in multiple parts of the app. The only difference between the two serializable classes is the approach to the values contained within – once as a file and once as a concrete class.

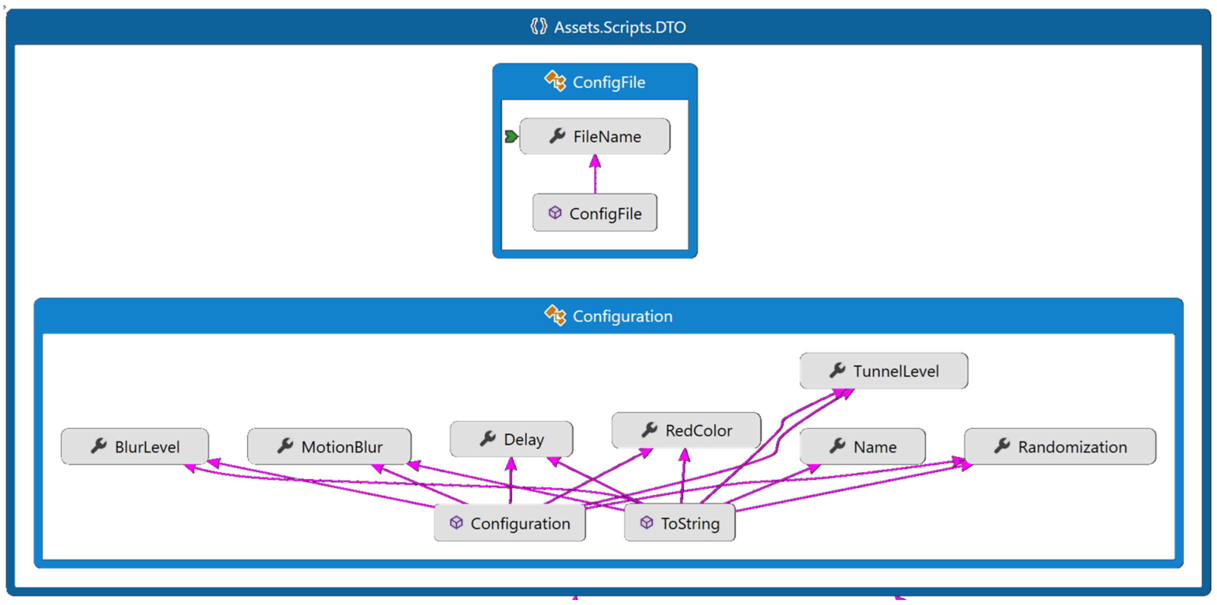


Figure 16: CodeMap DTO

8.3.3 Helper-Classes

The classes within this folder provide publicly accessible classes used to help with common calls to provide configurations and work with the values saved in the player preferences of the app.

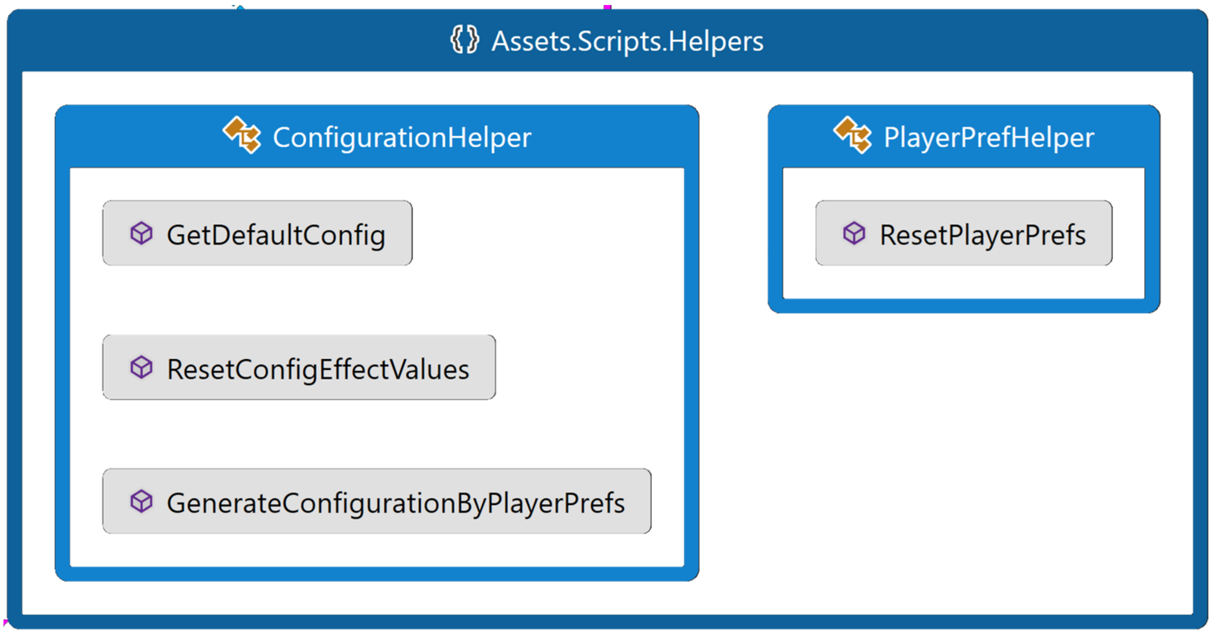


Figure 17: CodeMap Helpers

8.3.4 Metadata

All components within this folder are purely providing information and no functionality at all. One of the primary goals of this is the removal of magic numbers and strings within the code. This also helps with efficiency as well as the avoidance of typing errors due to autocompletion.

This unified access of any kind of static values also provides an easy entry point for future additions to any of the components such as the default configurations.

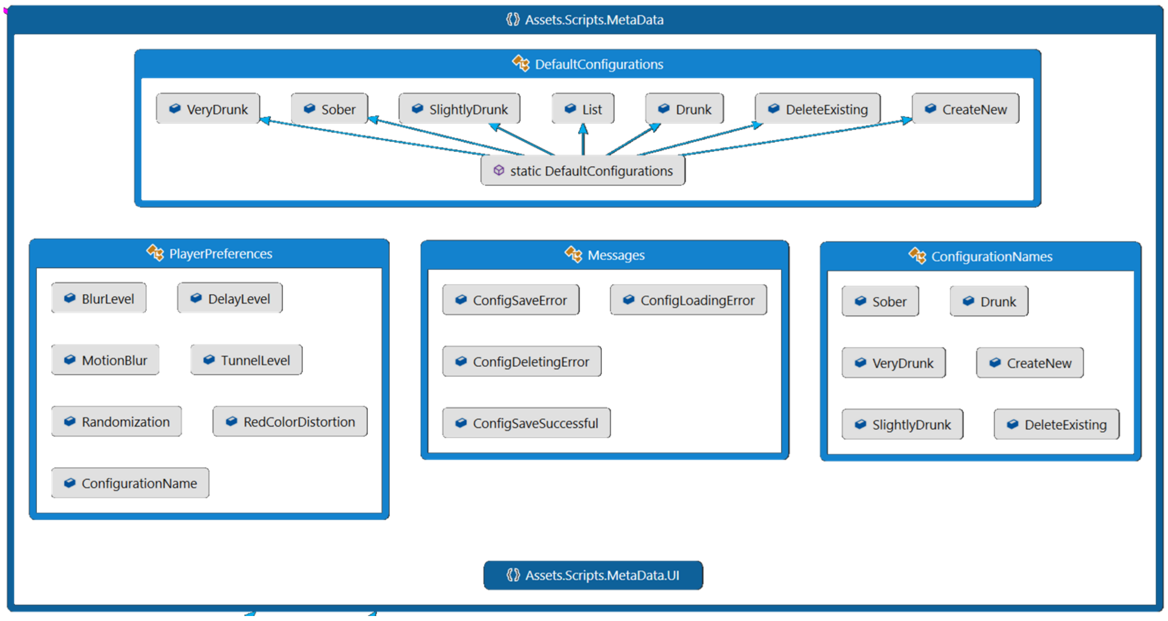


Figure 18: CodeMap Metadata

8.3.5 Metadata.UI

Since all UI components must be accessed via their name in Unity, the same principles used in the metadata namespace also apply here. The classes contained herein all relate to the individual scenes present in the Unity project and cover all the controls used within our custom code.

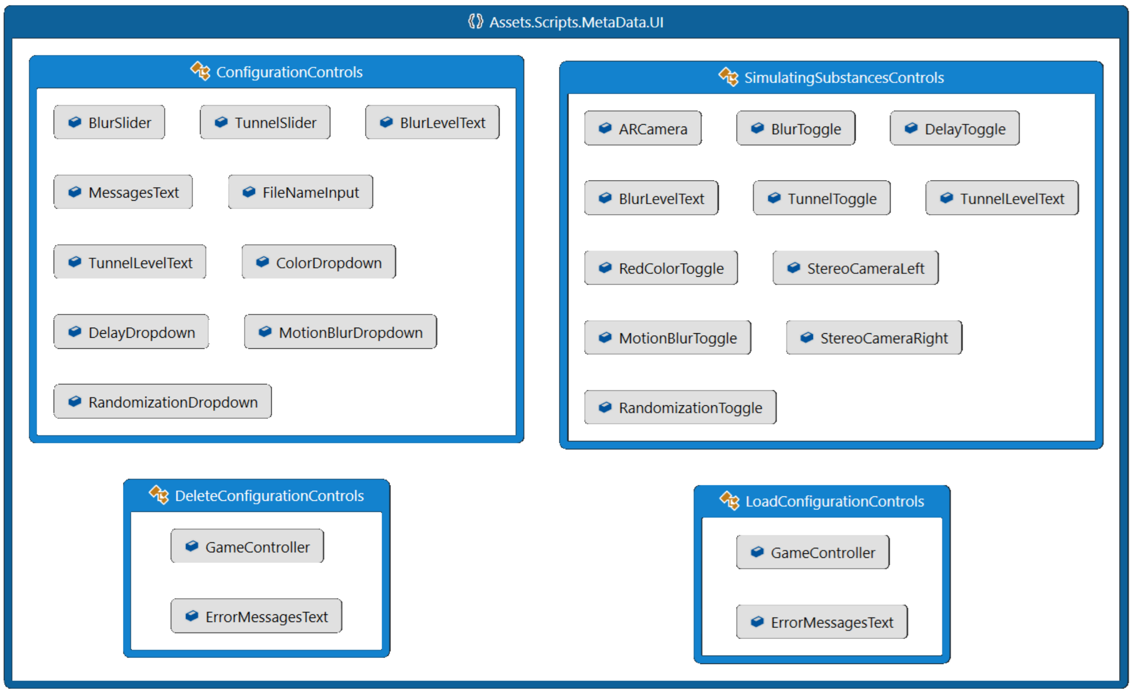


Figure 19: CodeMap Metadata.UI

8.4 Unity Project and the Graphical User Interface (GUI)

The image below provides an overview of the navigation in the Simulating Substances application. The entry point to the application is the Main scene:

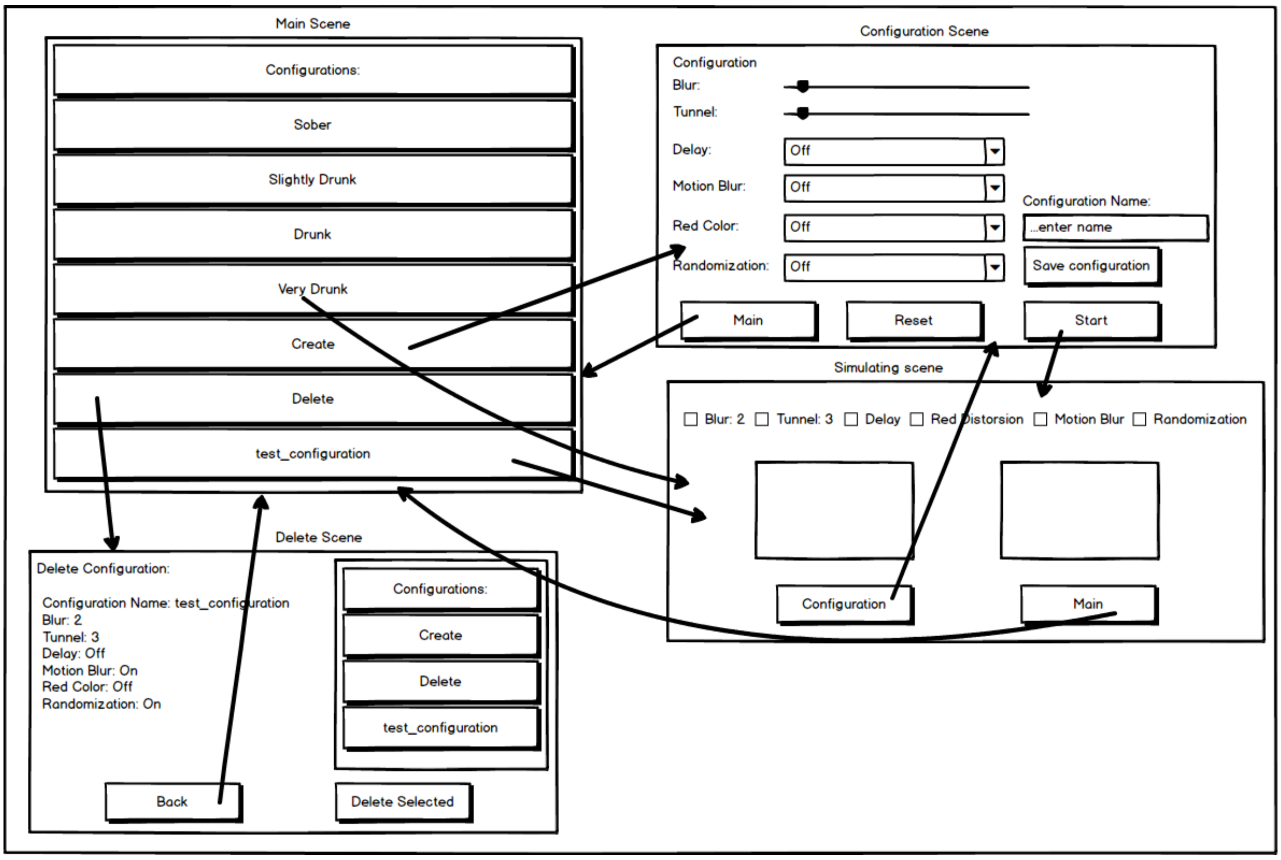


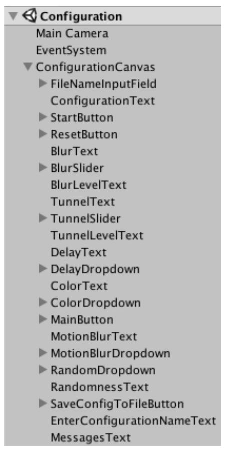
Figure 20: GUI Navigation Overview

Unity uses the term „Scene“ in order to refer to a file, which contains the objects of your game. Scenes can be used to create a main menu, individual levels, and anything else. Think of each unique Scene file as a unique level. In each Scene, you will place your environments, obstacles, and decorations, essentially designing and building your game in pieces.

The Hierarchy Window in Unity is a hierarchical text representation of every object in the scene. The hierarchy reveals the structure of how objects are attached to one another and can became very complex.

Below we explain the four Simulating Substances application scenes ordered by growing complexity. Each scene has a corresponding C# script responsible of its control following the model-view-controller(MVC) pattern.

For a better understanding of the GUI it’s necessary to explain before how is data preserve scenes.



```
▼ Configuration
    Main Camera
    EventSystem
  ▼ ConfigurationCanvas
    ▶ FileNameInputField
      ConfigurationText
    ▶ StartButton
    ▶ ResetButton
      BlurText
    ▶ BlurSlider
      BlurLevelText
      TunnelText
    ▶ TunnelSlider
      TunnelLevelText
      DelayText
    ▶ DelayDropdown
      ColorText
    ▶ ColorDropdown
    ▶ MainButton
      MotionBlurText
    ▶ MotionBlurDropdown
    ▶ RandomDropdown
      RandomnessText
    ▶ SaveConfigToFileButton
      EnterConfigurationNameText
      MessagesText
```

Abbildung 21: Configuration Scene Hierarchy

8.4.1 Communication between scenes

A scene is an independent entity with its own lifecycle. That means, that when a new scene is loaded (f. e. a new level in a game) all objects from the previous scene are destroyed.

In order to preserve data Unity provides the PlayerPrefs class, which stores and accesses player preferences between game sessions. This class uses a simple Key/Value system to save the data to a file:

```
PlayerPrefs.SetInt("Player Score", 10);

var score = PlayerPref.GetInt("Player Score")
```

We employ this class to preserve the configuration values of the image effects selected by the user so the user can switch between scenes or even shut down the application without losing the configuration.

8.4.2 Simulating Substances scene

This is the scene where all the action happens. The Vuforia plugin provides the AR functionality with the ARCamera component and the different image effects provided by Unity create the desired effects such as blur or tunnel view.

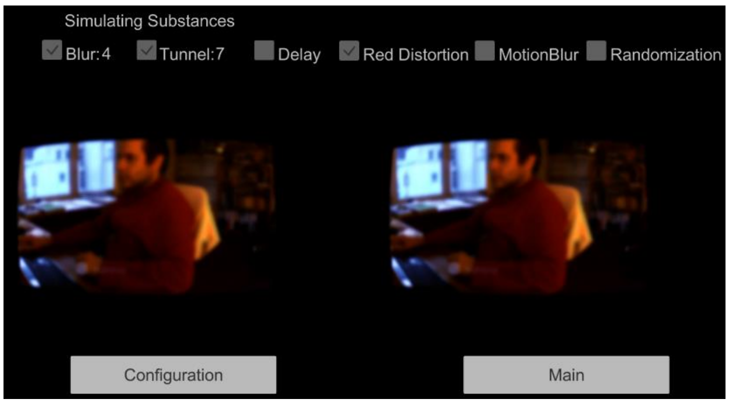


Figure 22: Simulating Substances Scene Preview

The effect values shown in the upper part are loaded from the PlayerPrefs when the scene starts. In the **SimulatingViewModel** class:

```
UpdateBlurValue(PlayerPrefs.GetFloat(PlayerPreferences.BlurLevel));
UpdateTunnelValue(PlayerPrefs.GetFloat(PlayerPreferences.TunnelLevel));
UpdateDelay(PlayerPrefs.GetInt(PlayerPreferences.DelayLevel));


        private void UpdateBlurValue(float value)
        {
            if (value > 0)
            {
                blurValueText.text = value.ToString(); //Shows the 4 in the image

                blurToggle.isOn = true;     //Shows     the     toggle     as     selected
            }
        }
```

As you can see in the image, this scene contains the application's main functionality; it shows the user the augmented reality.

Problems:

The main problem we discovered in this scene is the size of the augmented reality windows. The following two images provide a comparison of a virtual reality application (top) and the augmented reality application (bottom):

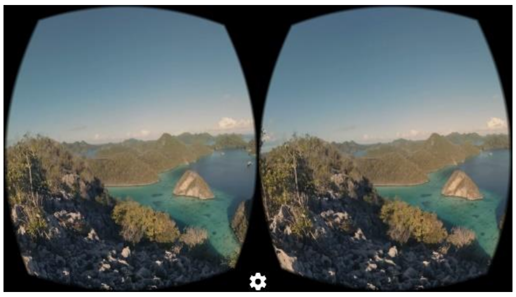


Figure 23: Virtual Reality Camera

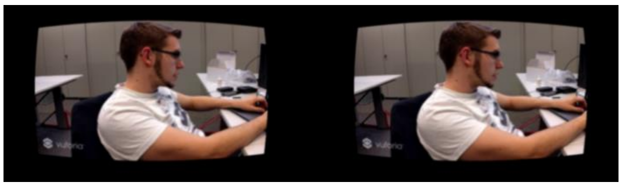
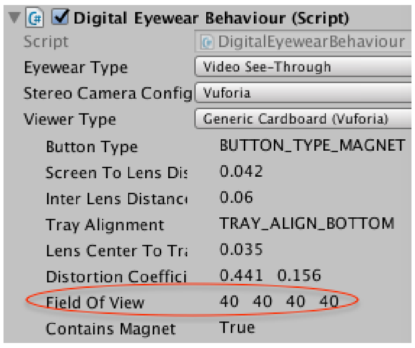


Figure 24: Augmented Reality Camera

Although the images are not of the same size, you can see that the size of the augmented reality windows is much smaller. After a rather intense research, we could determine that this is a limitation of the Vuforia Plugin used for augmented reality.

The Vuforia AR Camera only has a 40 degrees Field of View capacity and this condition of course modifies the windows size. The reason for this of course is also related to the image captured by the smartphone camera used by this component whereas in a VR app, the developer can choose to provide a better field of view.



Digital Eyewear Behaviour (Script)

Script	DigitalEyewearBehaviour
Eyewear Type	Video See-Through
Stereo Camera Config	Vuforia
Viewer Type	Generic Cardboard (Vuforia)
Button Type	BUTTON_TYPE_MAGNET
Screen To Lens Dis	0.042
Inter Lens Distanc	0.06
Tray Alignment	TRAY_ALIGN_BOTTOM
Lens Center To Tr	0.035
Distortion Coeffici	0.441 0.156
Field Of View	40 40 40 40
Contains Magnet	True

Figure 25: Vuforia AR Component Properties

8.4.3 Configuration scene

In this scene, you can activate or deactivate effects and configure the effects strength. The configuration is saved automatically to the PlayerPrefs. It is also possible to save a custom configuration to a file.

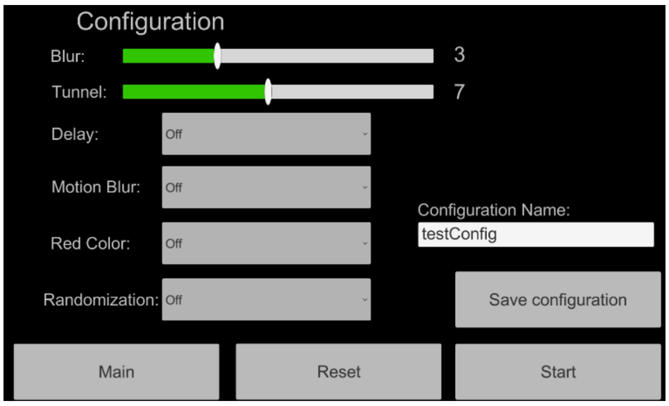

Configuration
Blur: 3
Tunnel: 7
Delay: Off
Motion Blur: Off
Red Color: Off
Randomization: Off
Configuration Name: testConfig
Save configuration
Main Reset Start

Figure 26: Configuration Scene Preview

Problems:

The Configuration scene has more GUI components and this increases automatically the logic complexity. The Dropdowns and sliders introduce configuration values, which have to be saved in the PlayerPrefs in order to be transmitted to the Simulating Substances scene. At the same time, when the user decides to reset these values, all GUI components must be reset to the default values.

Another responsibility of the Configuration Scene is saving the configuration values to a file, in order to allow the user to have different configurations, which can be later loaded or deleted as needed.

Solutions:

For the two different problems, we divided the responsibility into two different classes, the **`ConfigurationViewModel`** and the **`SaveConfigurationSceneViewModel`** classes.

When the user f. e. decides to activate the red color distortion, he/she selects "On" in the red color dropdown. This action generates an event captured by the Unity's event system and calls an "`OnValueChanged(int32 value)`" method.

The Inspector in Unity allows binding this method call to a script, which in this case is the **`ConfigurationViewModel`** class:

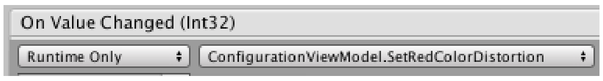

On Value Changed (Int32)
Runtime Only
ConfigurationViewModel.SetRedColorDistortion

Figure 27: Unity Method Call Binding

The `OnValueChanged(int32 value)` method calls the `SetRedColorDistortion` method of our **`ConfigurationViewModel`** class:

```
public void SetRedColorDistortion(int value)
{
    PlayerPrefs.SetInt(PlayerPreferences.RedColorDistortion, value);
}
```

In this method, we save the value (it just can be a 0 for off or a 1 for on) to the PlayerPrefs. And the same is done with all the GUI components.

When resetting the GUI components, first we must get these components from the scene hierarchy and then reset its values to the default values (usually 0).

We get the component from the hierarchy with:

```
Dropdown colorDropdown =

        GameObject.Find(ConfigurationControls.ColorDropdown).GetComponent<Dropdown>();
```

And then we reset its value to default: `colorDropdown.value = 0;`

This reset action is generated by the user when it presses the Reset button.

With the **`SaveConfigurationSceneViewModel`** class we address the file saving problem. The user selects the different configuration values in the configuration scene, enters the configuration's name in the input field and presses the "Save configuration" button.

This action invokes the `SaveConfiguration` method of the **`SaveConfigurationSceneViewModel`** class:

```
public void SaveConfiguration()
{
var fileNameInputField =

      GameObject.Find(ConfigurationControls.FileNameInput).GetComponent<InputField>();
SaveToFile(ConfigurationHelper.GenerateConfigurationByPlayerPrefs(fileNameInputField.text);

}
```

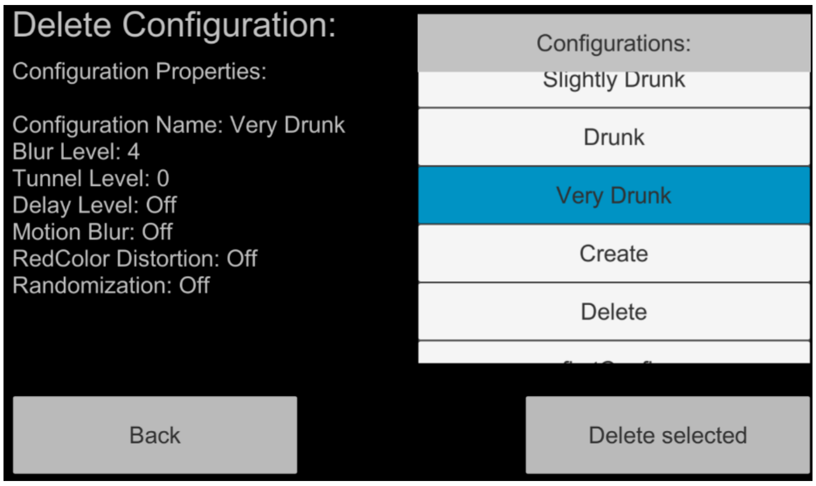
The `ConfigurationHelper.GenerateConfigurationByPlayerPrefs(fileNameInputField.text)`method is a helper method, which creates a `Configuration` object from the already saved data in the PlayerPrefs and passes it to the `SaveToFile` function.

This `SaveToFile` method just calls another `Save` helper method responsible for the file saving and takes care of any exception occurring:

```
public static void Save(Configuration configuration)
{
   var bf = new BinaryFormatter();
   var file = File.Create(Application.persistentDataPath + "/" + configuration.Name)

    bf.Serialize(file, configuration);
    file.Close();
}
```

8.4.4 Delete Scene

In this scene, the user is able to delete the custom configurations saved before in the configuration scene. He/she can also see the different values of the configurations he created.



Delete Configuration:

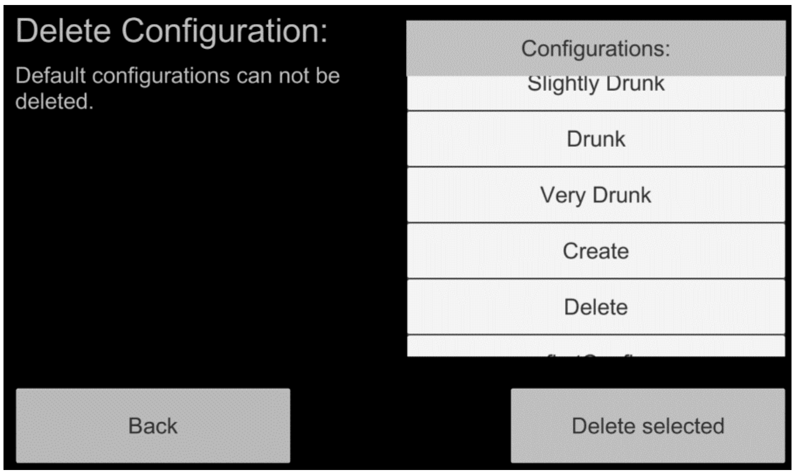
Configuration Properties:

Configuration Name: Very Drunk
Blur Level: 4
Tunnel Level: 0
Delay Level: Off
Motion Blur: Off
RedColor Distortion: Off
Randomization: Off

Figure 28: Delete Scene Preview

Problems:

This scene contains a scrollable list with all the configurations, both the default and custom configurations. From this list the user selects a configuration item and sees its values and deletes it if desired. If the selected item is a default configuration an error message is shown:



Delete Configuration:

Default configurations can not be deleted.

Figure 29: Delete Scene with Error Message

The problem is that Unity does not have standard GUI Elements like scrollable lists with selectable items. They have to be built from the given elements (Buttons, Fields, Panels, etc.) and filled dynamically at run time.

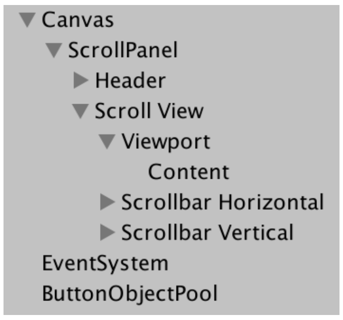
Solutions:

After doing some more research, we found several tutorials, which address this problem with the Unity tutorial[4] being the most appropriate for our needs.

This tutorial makes a scrollable list with buttons as list items starting from a panel with following features:

- The scrollable panel automatically scales to fit the content
- The content (buttons in this case) have their preferred sizes
- The content is clipped/masked using an image mask
- Vertical Scroll

The object's hierarchy tree eventually looks like:



```
▼ Canvas
  ▼ ScrollPanel
    ▶ Header
    ▼ Scroll View
      ▼ Viewport
          Content
      ▶ Scrollbar Horizontal
      ▶ Scrollbar Vertical
  EventSystem
  ButtonObjectPool
```

Figure 30: Scrollable List Hierarchy

Here's what each object does and the components it needs:

Canvas[5]

The Canvas is the area that all UI elements should be inside. The Canvas is a Game Object with a Canvas component on it, and all UI elements must be children of such a Canvas.

Scroll Panel

This is just the parent Game Object containing the other elements

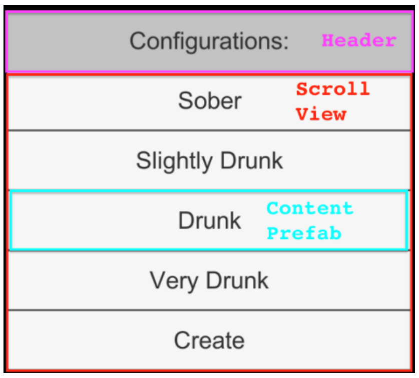


Abbildung 31: Scrollable List Preview with Comments

Header

A button without functionality

Scroll View

Contains elements that set up how the content is viewed. The **Viewport** is the component that actually contains the list elements. Both **Horizontal** and **Vertical scrollbars** define the list's behavior when the user scrolls up or down the list. In this case, only vertical scrolling is allowed of course.

[4] https://unity3d.com/learn/tutorials/topics/user-interface-ui/intro-and-setup?playlist=17111
[5] https://docs.unity3d.com/Manual/UICanvas.html

Viewport Content: Button Prefabs

The list will be filled on runtime with default configuration and custom configurations. That means that the length and size of the list is unknown until the scene is loaded and that we have to generate the list items and add them to the list dynamically.

For this we use Unity prefabs[6], a game object template from which you can create new object instances in the scene.

We built a button prefab with our desired features, and we instantiate them like:

```
public GameObject listButton; // Button Prefab

                      […]

private void FillListInGui()
{
 foreach (var file in configFilesList)
                                                                                    {
     var newButton = Instantiate(listButton);                 // Instantiates  Button Prefab
    var button = newButton.GetComponent<ListButton>();//   Button   contains   a   C#   class
    button.nameLabel.text = file.FileName;    // New    button    shows    the    file    name
    var index = configFilesList.IndexOf(file);
    button.button.onClick.AddListener(() => { OnButtonClicked(index); });
    newButton.transform.SetParent(contentPanel); // Adds new Button to parent panel in

                                                 //   order   to   keep   the   hierarchy   tree
     }
}
```

The advantages of prefabs are obvious. You are able to reuse the prefab in different scenes in combination with different game objects and settings.

For more detailed information about the scrollable list with selectable items please refer to the above-mentioned Unity tutorial.

8.4.5 Main Scene

The main scene is the entry point to the Simulating Substances application. From this scene, the user can create or delete configurations, start from a default configuration or start from a custom configuration.

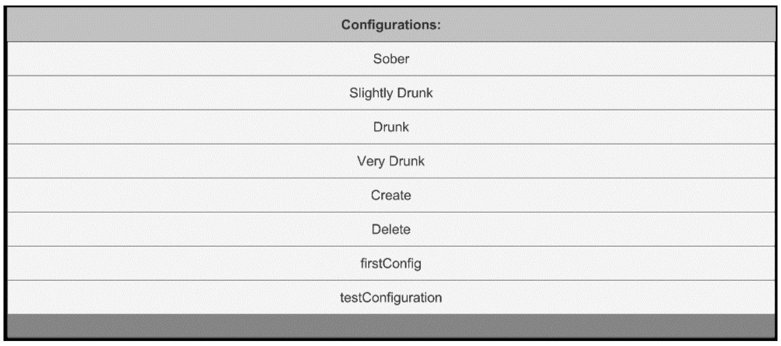


Figure 32: Main Scene Preview

[6] https://docs.unity3d.com/Manual/Prefabs.html

Problems:

Just like in the delete scene, the first problem was to implement a scrollable list with selectable items again. Please refer to the solutions in the delete scene section to see how that has been solved.

Another unexpected problem in this scene we discovered is that the application’s event processing does not work properly when the application is running on a mobile phone. When the user tries to select a list item by clicking on it, sometimes the first click gets lost and the selection does not work as intended. This behavior is not deterministic, so sometimes it works and other times it does not.

A rather long research revealed no documentation about this issue and the only preliminary conclusion we could come to is, that it is a still not documented bug in Unity or our test devices simply malfunction.

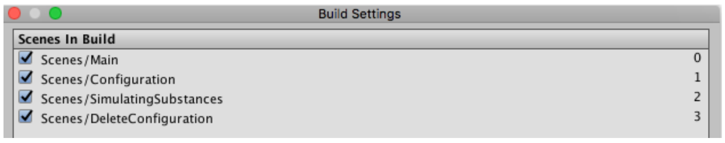
8.4.6 Navigation between scenes

As our application has a simple structure with only 4 scenes we looked for a simple navigation solution too.

Unity offers the `SceneManager`[7], which manages the scenes at run-time and provides the following method:

```
private void LoadScene(int sceneIndex)
{
    SceneManager.LoadScene(sceneIndex);
}
```

Each scene has an index in the Unity Editor:



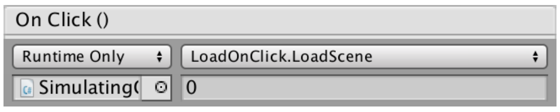
Build Settings

Scenes In Build

Scenes/Main	0
Scenes/Configuration	1
Scenes/SimulatingSubstances	2
Scenes/DeleteConfiguration	3

Figure 33: Unity Build Settings with Scene Index

So, whenever we need to navigate to another scene, we call the LoadScene method from a GUI component (e.g. a button) with the corresponding index:



On Click ()

Runtime Only | LoadOnClick.LoadScene

Simulating(| 0

Figure 34: Button Click Event linked to Load Scene

[7] https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html

9 Challenges

This chapter provides a better insight into the challenges we have come across throughout this project.

9.1 Differentiation between errors, bugs and limitations

One of the biggest challenges coming with the experimental focus of this project was the differentiation between encountering bugs and limitations. Due to our lack of previous experience with the used technologies, we were typically amazed how fast we could get something to run. But when that was not the case or the working feature had a small issue/bug, we basically always had to spend an exceptional amount of time on understanding why it was caused and attempting to fix it.

Some example encounters of the issue described above include the following:

- The “Click not registering” issue on UI controls with still no applicable solution
- The reason why we had to create our own version of the TiltShift component using the existing and, unmodified code because we had limited access to its properties otherwise
- The reason why we were unable to use the inherited “enabled” property on our custom delay component, even though this was what we used for all other components without any issues
- The reason why our delay component manages to crash most phones on most occasions when combined with other image effects. Even after talking to the Unity customer support and finding out, that the crash is eventually caused by the VuforiaWrapper trying to access something it cannot.

9.2 Performance Optimization [14]

Even at the beginning of the development process while still experimenting with the different image effects, the smartphones used for testing seemed to reach their limit regarding performance/processing power. Since we were unaware of whether that is caused by our app, custom code or simply to be expected using the AR components at our hands, we ordered and read a book specifically written for this called “Unity 5 Game Optimization” (see reference for this chapter).

Unfortunately, the only techniques useful for our custom code within the Virtual Buzz app and elaborated in this book can roughly be explained in the next small chapters. Due to the simplicity of the apps code, there sadly has not been a big performance achievement caused by those changes, but it sure helped with the overall understanding of Unity and was a good read.

9.2.1 Caching Component references

All elements in Unity are commonly accessed through the `GameObject.Find(“COMPONENTNAME”).GetComponent<T>();` command and this is said to be a very costly call especially if it is done over and over again. So, in order to save performance here this sort of call should never be used in very frequently occurring calls and instead be done once in the initialization and then cached by reference.

9.2.2 Removing empty callbacks

Upon creating new scripts which are derived from MonoBehaviour, there are two method templates added by default:

```
// Use this for initialization
void Start () {
}
// Update is called once per frame
void Update () {
}
```

Even if these declarations stay empty, they will be added to all the other methods of this type upon initialization and therefore do have a performance impact.

9.3 Hardware limitations

As already indicated by the previous chapter, we have managed to identify quite a bit of hardware limitations when it comes to smartphones based AR experiences. In order to give a bit of an insight on the performance of the smartphones used for testing and perhaps provide proof, that the developed app is expected to be more enjoyable on newer smartphones (optimized for AR/VR in the best case) as well as the future, we have gathered the profiling data of three devices. These devices are two smartphones and a personal computer and all of these are using the same configuration in an attempt to compare apples to apples. The configuration used for the Virtual Buzz app is as follows:

• Blur:	5
• TiltShift:	7
• RedColorDistortion:	On
• MotionBlur:	On
• Randomization:	On
• Delay:	Off

What also needs to be mentioned is, that having the Unity profiler active on a device while running the app of course comes with a rather significant impact on the performance due to the generated overhead. However, the results provided are not meant to give exact value representations but a rough insight into how well the devices at hand can handle the workload caused by the app.

The profiling screenshots provided cover the CPU and memory area on all devices and on the computer also the GPU component, since that is the only instance where it was available for profiling. On each of the profiles captured, we selected one of the peaks of medium height (white bar) regardless of whether that was representative of the average performance. In the CPU and GPU areas of the individual profiles, we highlighted the influence of the custom components used by our app on the performance in an attempt to proof, that these are not the primary cause of the performance struggle.

The results clearly suggest, that although our app is not handled extremely well on the smartphones used during the development of this project, it has not been a challenge for a rather powerful computer at all. We thus feel reinforced to say our claim, that the developed app will continue to provide a better performance as it is used on newer smartphones with better components in the future, has been proofed to a sufficient extent.

9.3.1 Nexus 5X

This smartphone was considered a mid-range phone when released in October 2015, provided by the HSR and served as our primary test subject since our industrial partner also primarily used the same device.

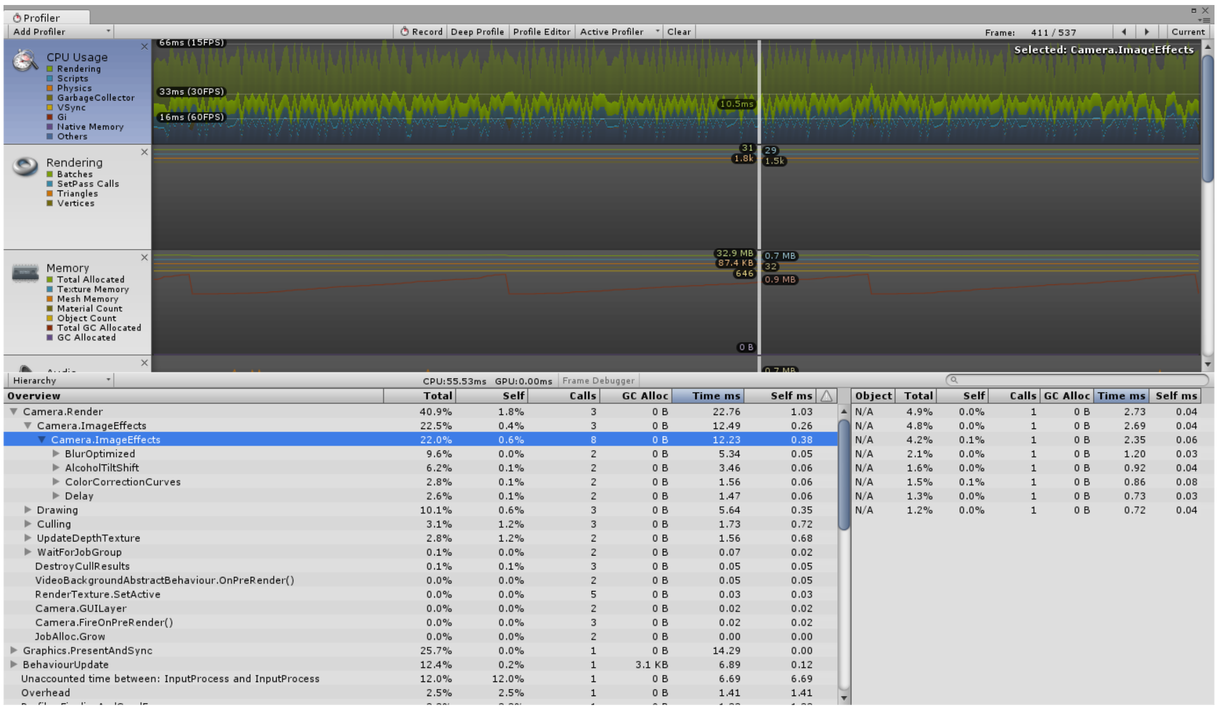
Because of the focus on this device, we have also gathered the following observational data on the usage of the app over an extended period of time:

Time passed	**Battery consumption**	**Heat impression**
10 min	5%	noticeable heat around camera
15 min	8%	very noticeable heat around camera
20 min	10%	very noticeable heat across the upper parts of the phone
25 min	16%	concerning heat across the upper parts of the phone

Table 6: Nexus 5X App Usage Observation

Unfortunately, we have not found any concrete information on whether the reached temperatures are harmful to the device and should be avoided at all cost, or still considered safe and can be ignored. This data does lead us to the hypothesis though, that this smartphone is struggling with handling the generated workload and extended use may not be advisable.

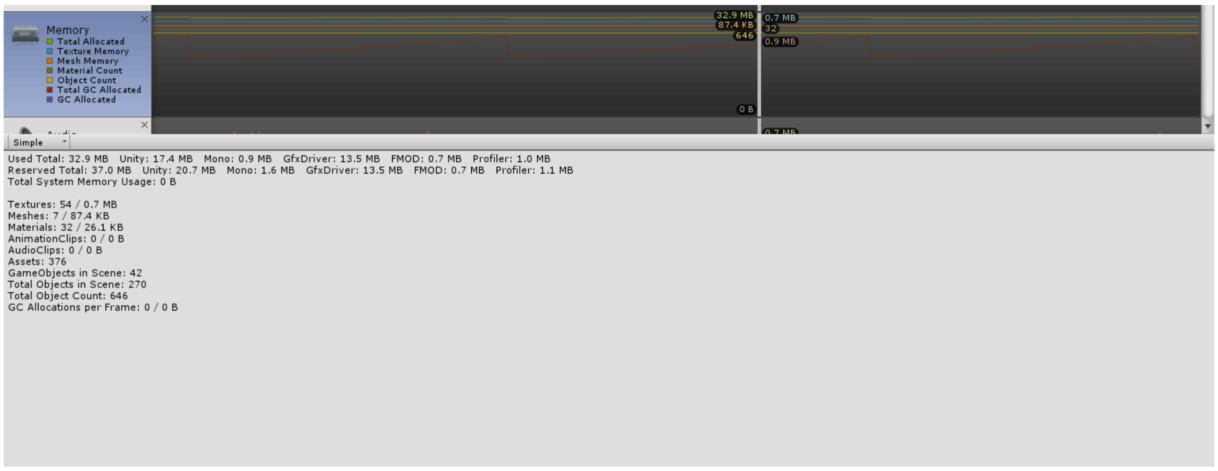
The CPU profiling performed further established the mentioned hypothesis, since the frames per second (FPS) displayed on the smartphone are much closer to 15 FPS, which our eyes do not recognize as flued, than anything else.



Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
▼ Camera.Render	40.9%	1.8%	3	0 B	22.76	1.03
▼ Camera.ImageEffects	22.5%	0.4%	3	0 B	12.49	0.26
▼ Camera.ImageEffects	22.0%	0.6%	8	0 B	12.23	0.38
▶ BlurOptimized	9.6%	0.0%	2	0 B	5.34	0.05
▶ AlcoholTiltShift	6.2%	0.1%	2	0 B	3.46	0.06
▶ ColorCorrectionCurves	2.8%	0.1%	2	0 B	1.56	0.06
▶ Delay	2.6%	0.1%	2	0 B	1.47	0.06
▶ Drawing	10.1%	0.6%	3	0 B	5.64	0.35
▶ Culling	3.1%	1.2%	3	0 B	1.73	0.72
▶ UpdateDepthTexture	2.8%	1.2%	2	0 B	1.56	0.68
▶ WaitForJobGroup	0.1%	0.0%	2	0 B	0.07	0.02
DestroyCullResults	0.1%	0.1%	3	0 B	0.05	0.05
VideoBackgroundAbstractBehaviour.OnPreRender()	0.0%	0.0%	2	0 B	0.05	0.05
RenderTexture.SetActive	0.0%	0.0%	5	0 B	0.03	0.03
Camera.GUILayer	0.0%	0.0%	2	0 B	0.02	0.02
Camera.FireOnPreRender()	0.0%	0.0%	3	0 B	0.02	0.02
JobAlloc.Grow	0.0%	0.0%	2	0 B	0.00	0.00
▶ Graphics.PresentAndSync	25.7%	0.0%	1	0 B	14.29	0.00
▶ BehaviourUpdate	12.4%	0.2%	1	3.1 KB	6.89	0.12
Unaccounted time between: InputProcess and InputProcess	12.0%	12.0%	1	0 B	6.69	6.69
Overhead	2.5%	2.5%	1	0 B	1.41	1.41

Object	Total	Self	Calls	GC Alloc	Time ms	Self ms
N/A	4.9%	0.0%	1	0 B	2.73	0.04
N/A	4.8%	0.0%	1	0 B	2.69	0.04
N/A	4.2%	0.1%	1	0 B	2.35	0.06
N/A	2.1%	0.0%	1	0 B	1.20	0.03
N/A	1.6%	0.0%	1	0 B	0.92	0.04
N/A	1.5%	0.1%	1	0 B	0.86	0.08
N/A	1.3%	0.0%	1	0 B	0.73	0.03
N/A	1.2%	0.0%	1	0 B	0.72	0.04

Figure 35: CPU Profiling Nexus 5X



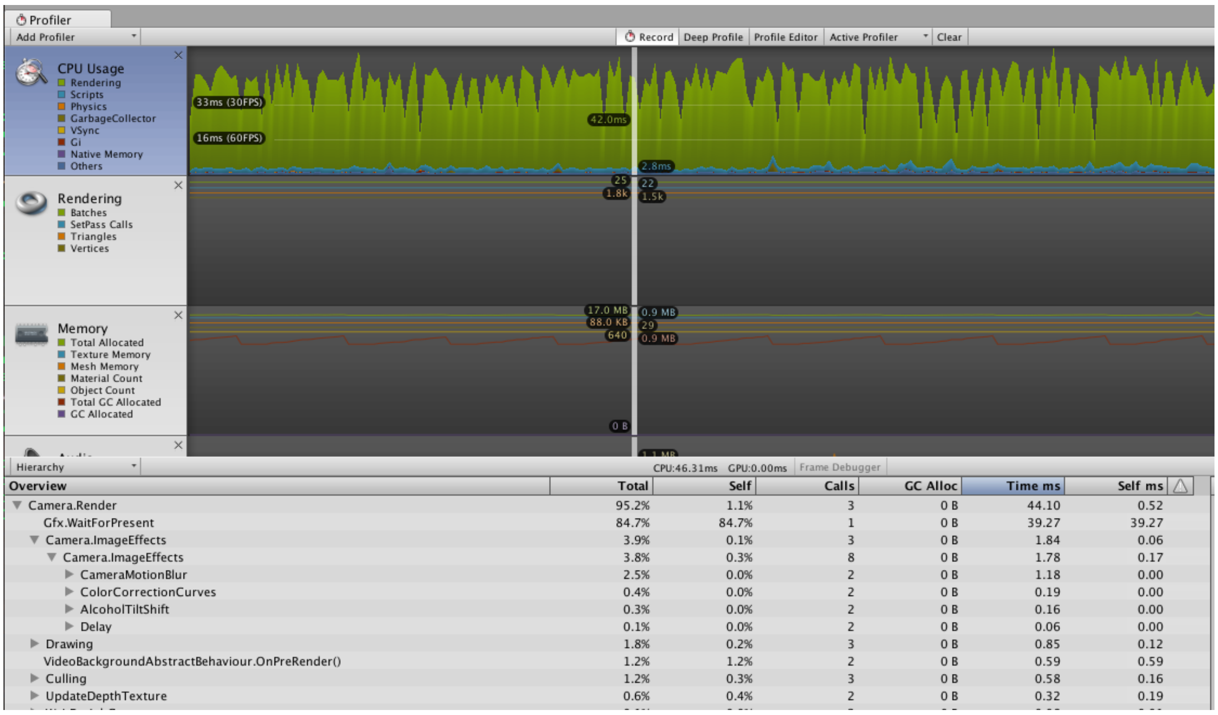
```
Used Total: 32.9 MB   Unity: 17.4 MB   Mono: 0.9 MB   GfxDriver: 13.5 MB   FMOD: 0.7 MB   Profiler: 1.0 MB
Reserved Total: 37.0 MB   Unity: 20.7 MB   Mono: 1.6 MB   GfxDriver: 13.5 MB   FMOD: 0.7 MB   Profiler: 1.1 MB
Total System Memory Usage: 0 B

Textures: 54 / 0.7 MB
Meshes: 7 / 87.4 KB
Materials: 32 / 26.1 KB
AnimationClips: 0 / 0 B
AudioClips: 0 / 0 B
Assets: 376
GameObjects in Scene: 42
Total Objects in Scene: 270
Total Object Count: 646
GC Allocations per Frame: 0 / 0 B
```

Figure 36: Memory Profiling Nexus 5X

9.3.2 iPhone 6 Plus

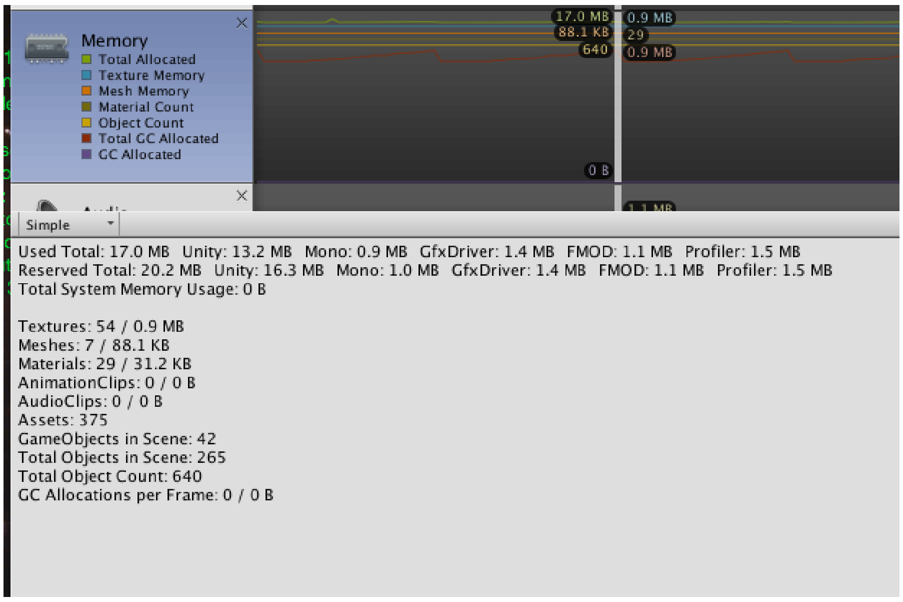
The iPhone 6 Plus was considered a flagship phone at the time of its release, but has been released slightly more than a year before the Nexus 5X in September 2014. This smartphone served us as the second testing subject, since its performance greatly surpassed the other available option at hand, the HTC A9.

The performance appears to be slightly better than the one seen on the Nexus 5X with the average FPS being somewhere near the middle between 15 and 30 FPS. This makes the experience appear rather fluent, but still with a noticeable stutter at times. It is also very interesting, that the memory used by the app is almost halved compared to the values of the Nexus 5X with a big difference in the space used by the GfxDriver.



Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
▼ Camera.Render	95.2%	1.1%	3	0 B	44.10	0.52
Gfx.WaitForPresent	84.7%	84.7%	1	0 B	39.27	39.27
▼ Camera.ImageEffects	3.9%	0.1%	3	0 B	1.84	0.06
▼ Camera.ImageEffects	3.8%	0.3%	8	0 B	1.78	0.17
▶ CameraMotionBlur	2.5%	0.0%	2	0 B	1.18	0.00
▶ ColorCorrectionCurves	0.4%	0.0%	2	0 B	0.19	0.00
▶ AlcoholTiltShift	0.3%	0.0%	2	0 B	0.16	0.00
▶ Delay	0.1%	0.0%	2	0 B	0.06	0.00
▶ Drawing	1.8%	0.2%	3	0 B	0.85	0.12
VideoBackgroundAbstractBehaviour.OnPreRender()	1.2%	1.2%	2	0 B	0.59	0.59
▶ Culling	1.2%	0.3%	3	0 B	0.58	0.16
▶ UpdateDepthTexture	0.6%	0.4%	2	0 B	0.32	0.19

Figure 37: CPU Profiling iPhone 6 Plus



```
Used Total: 17.0 MB   Unity: 13.2 MB   Mono: 0.9 MB   GfxDriver: 1.4 MB   FMOD: 1.1 MB   Profiler: 1.5 MB
Reserved Total: 20.2 MB   Unity: 16.3 MB   Mono: 1.0 MB   GfxDriver: 1.4 MB   FMOD: 1.1 MB   Profiler: 1.5 MB
Total System Memory Usage: 0 B

Textures: 54 / 0.9 MB
Meshes: 7 / 88.1 KB
Materials: 29 / 31.2 KB
AnimationClips: 0 / 0 B
AudioClips: 0 / 0 B
Assets: 375
GameObjects in Scene: 42
Total Objects in Scene: 265
Total Object Count: 640
GC Allocations per Frame: 0 / 0 B
```

Figure 38: Memory Profiling iPhone 6 Plus

9.3.3 Personal Computer

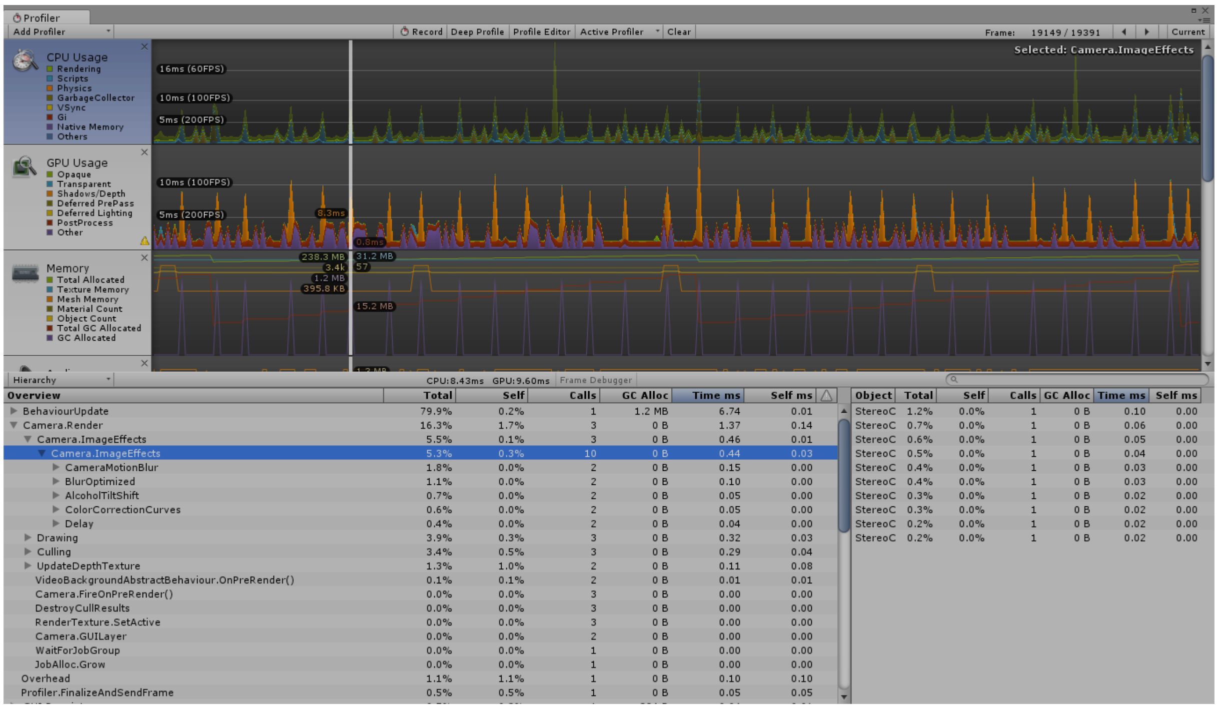
Since both of the phones used in this comparison had rather obvious issues with handling the Virtual Buzz app and there were none of the newest phones released such as the Google Pixel at our disposal, we felt like we had to provide another high-performance input with a vast difference to the other contenders to reinforce our point.

Hardware specifications:

- CPU: Intel Core i7 980, 6 cores @ 3.33 GHz
- GPU: ZOTAC GeForce GTX 1080, 2560 Cores @ 1911 MHz, 8 GB GDDR5X
- using a 720p Webcam with a quality setting of 8 Megapixels

The components inside this computer can currently can be seen as somewhere in the medium to upper range of high end computers. The graphics card is among the newest ones available and uses the most up to date architecture available and is optimized for VR applications whereas the CPU (and rest of the internals) are outdated by a couple of years.

The drastic difference to the previous profiling results is about as obvious as it gets. The experience is absolutely fluent and smooth with a few major drops in the frames per second related to the garbage collector among other things. The additional information available on the GPU profiling segment also clearly shows, that most of the workload is not caused by the used image effects, but the continuous capture of the camera and possibly more of the internals within the ARCamera by Vuforia associated with it.

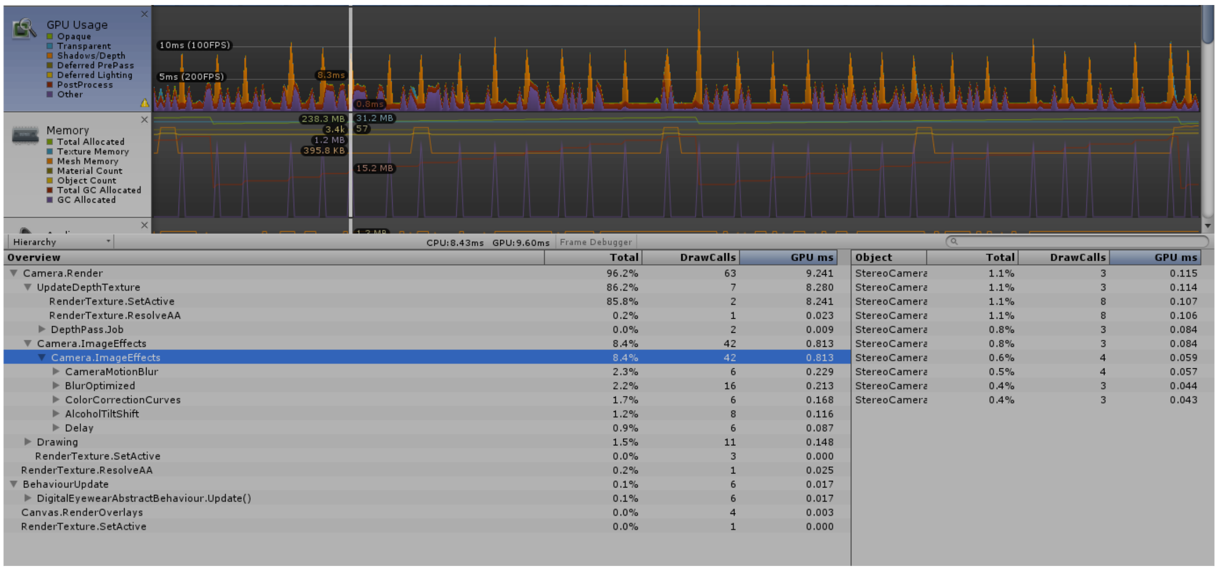


CPU:8.43ms GPU:9.60ms

Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
▶ BehaviourUpdate	79.9%	0.2%	1	1.2 MB	6.74	0.01
▼ Camera.Render	16.3%	1.7%	3	0 B	1.37	0.14
▼ Camera.ImageEffects	5.5%	0.1%	3	0 B	0.46	0.01
▼ Camera.ImageEffects	5.3%	0.3%	10	0 B	0.44	0.03
▶ CameraMotionBlur	1.8%	0.0%	2	0 B	0.15	0.00
▶ BlurOptimized	1.1%	0.0%	2	0 B	0.10	0.00
▶ AlcoholTiltShift	0.7%	0.0%	2	0 B	0.05	0.00
▶ ColorCorrectionCurves	0.6%	0.0%	2	0 B	0.05	0.00
▶ Delay	0.4%	0.0%	2	0 B	0.04	0.00
▶ Drawing	3.9%	0.3%	3	0 B	0.32	0.03
▶ Culling	3.4%	0.5%	3	0 B	0.29	0.04
▶ UpdateDepthTexture	1.3%	1.0%	2	0 B	0.11	0.08
VideoBackgroundAbstractBehaviour.OnPreRender()	0.1%	0.1%	2	0 B	0.01	0.01
Camera.FireOnPreRender()	0.0%	0.0%	3	0 B	0.00	0.00
DestroyCullResults	0.0%	0.0%	3	0 B	0.00	0.00
RenderTexture.SetActive	0.0%	0.0%	3	0 B	0.00	0.00
Camera.GUILayer	0.0%	0.0%	2	0 B	0.00	0.00
WaitForJobGroup	0.0%	0.0%	1	0 B	0.00	0.00
JobAlloc.Grow	0.0%	0.0%	1	0 B	0.00	0.00
Overhead	1.1%	1.1%	1	0 B	0.10	0.10
Profiler.FinalizeAndSendFrame	0.5%	0.5%	1	0 B	0.05	0.05

Object	Total	Self	Calls	GC Alloc	Time ms	Self ms
StereoC	1.2%	0.0%	1	0 B	0.10	0.00
StereoC	0.7%	0.0%	1	0 B	0.06	0.00
StereoC	0.6%	0.0%	1	0 B	0.05	0.00
StereoC	0.5%	0.0%	1	0 B	0.04	0.00
StereoC	0.4%	0.0%	1	0 B	0.03	0.00
StereoC	0.4%	0.0%	1	0 B	0.03	0.00
StereoC	0.3%	0.0%	1	0 B	0.02	0.00
StereoC	0.3%	0.0%	1	0 B	0.02	0.00
StereoC	0.2%	0.0%	1	0 B	0.02	0.00
StereoC	0.2%	0.0%	1	0 B	0.02	0.00

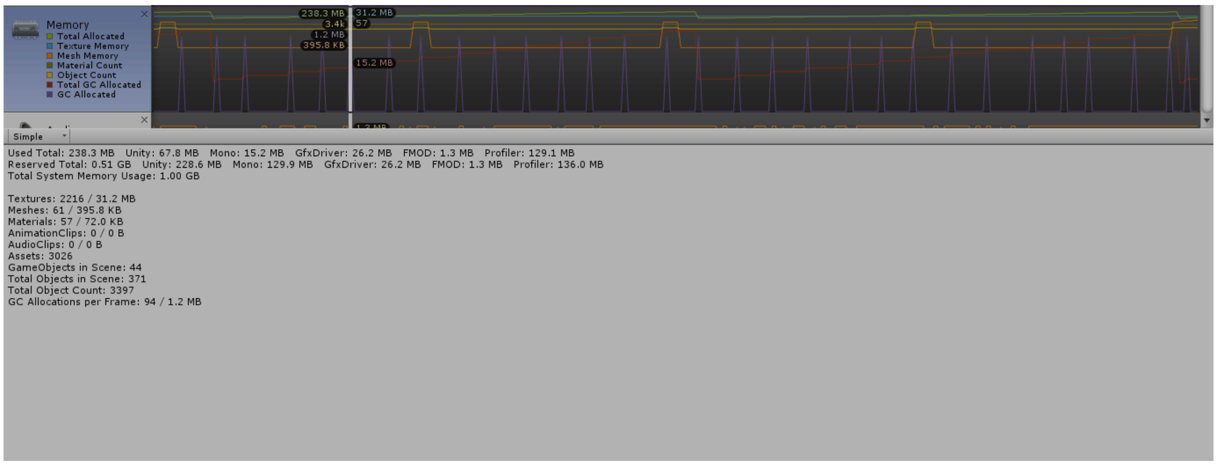
Figure 39: PC Profiling CPU



Overview	Total	DrawCalls	GPU ms
Camera.Render	96.2%	63	9.241
UpdateDepthTexture	86.2%	7	8.280
RenderTexture.SetActive	85.8%	2	8.241
RenderTexture.ResolveAA	0.2%	1	0.023
DepthPass.Job	0.0%	2	0.009
Camera.ImageEffects	8.4%	42	0.813
Camera.ImageEffects	8.4%	42	0.813
CameraMotionBlur	2.3%	6	0.229
BlurOptimized	2.2%	16	0.213
ColorCorrectionCurves	1.7%	6	0.168
AlcoholTiltShift	1.2%	8	0.116
Delay	0.9%	6	0.087
Drawing	1.5%	11	0.148
RenderTexture.SetActive	0.0%	3	0.000
RenderTexture.ResolveAA	0.2%	1	0.025
BehaviourUpdate	0.1%	6	0.017
DigitalEyewearAbstractBehaviour.Update()	0.1%	6	0.017
Canvas.RenderOverlays	0.0%	4	0.003
RenderTexture.SetActive	0.0%	1	0.000

Object	Total	DrawCalls	GPU ms
StereoCamera	1.1%	3	0.115
StereoCamera	1.1%	3	0.114
StereoCamera	1.1%	8	0.107
StereoCamera	1.1%	8	0.106
StereoCamera	0.8%	3	0.084
StereoCamera	0.8%	3	0.084
StereoCamera	0.6%	4	0.059
StereoCamera	0.5%	4	0.057
StereoCamera	0.4%	3	0.044
StereoCamera	0.4%	3	0.043

Figure 40: PC Profiling GPU



```
Used Total: 238.3 MB   Unity: 67.8 MB   Mono: 15.2 MB   GfxDriver: 26.2 MB   FMOD: 1.3 MB   Profiler: 129.1 MB
Reserved Total: 0.51 GB   Unity: 228.6 MB   Mono: 129.9 MB   GfxDriver: 26.2 MB   FMOD: 1.3 MB   Profiler: 136.0 MB
Total System Memory Usage: 1.00 GB

Textures: 2216 / 31.2 MB
Meshes: 61 / 395.8 KB
Materials: 57 / 72.0 KB
AnimationClips: 0 / 0 B
AudioClips: 0 / 0 B
Assets: 3026
GameObjects in Scene: 44
Total Objects in Scene: 371
Total Object Count: 3397
GC Allocations per Frame: 94 / 1.2 MB
```

Figure 41: PC Profiling Memory

10 Game Ideas

We tried to divide the different game ideas in categories for a more detailed explanation. Each extension or game idea will then be evaluated with the following criteria:

What	Description	Values
Area:	Which area is the focus of this idea?	User Immersion Simulation Quality Utility Fun
Pros:	Which positive aspects does this idea come with?	
Contras:	Which negative aspects does this idea come with?	
Impact:	How big is the expected impact of this feature on the end user experience if it were to be implemented?	Low Medium High
Effort:	How much effort and resources would approximately be needed to implement this idea?	Low (less than a day) Medium (more than a day, but less than a week) High (about a week) Very high (more than a week)
Technological risk:	What is the estimation of the likelihood of unknown problems, performance limitations or the possibility of it not being achievable at all?	Low Medium High Very high

Table 7: Game Idea Evaluation Table

10.1 New features for the existing app

The following concrete ideas for new features in the existing app have already been worked out and are to be considered to be implemented in a future version.

10.1.1 Inconstancy improved with focused view

As long as the person remains still, without making vehement movements with the body or head, the blur level should decrease or even disappear. As soon as the person starts moving or shaking its head, the blur levels increase again, resulting in the vision becoming blurred and perhaps even causing nausea or dizziness.

This can be interpreted as the counterpart to the camera motion blur effect currently available. This could be implemented by using the phone’s accelerometer and gyroscope sensors to detect and quantify the movements and having those influence the used values in the activated image effects.

Area:	Simulation quality
Pros:	Simplicity of implementation Realism provided by the Effect
Contras:	Usage of sensors in addition to the camera could lead to additional performance issues
Impact:	High
Effort:	Medium
Technological risk:	Low

Table 8: Focused View Feature Evaluation

10.1.2 Evening Simulation

Alcohol levels increase steadily over time: You start sober at the beginning of the evening and as time and drinks pass throughout the evening you become more and more drunk.

Alcohol levels decrease steadily over time: Simulating the process of slowly getting better after a nice evening with lots of drinking.

This could very easily be implemented since is essentially a simplification (one-way) of the Randomization component (two-way) already in use and act as a utility improvement, since it can improve the time a phone can remain inside a VR headset without needing its configuration to be adjusted.

Area:	Utility
Pros:	Simplicity of implementation
Contras:	May help with frequency of configuration adjustments
Impact:	Medium
Effort:	Low
Technological risk:	Low

Table 9: Evening Simulation Feature Evaluation

10.2 Application of the Virtual Buzz App in real world activities

Based on the feedback provided by ASN after our last presentation of the app right before the end of our project, we tried to describe possible applications of the Virtual Buzz app in real world activities used to spread the message against the combination of driving in combination with the influence of alcohol or drugs.

10.2.1 On the way home

In this game a small group of young people and one or more advisors are involved. The group's goal is to arrive to the home of someone's group member by foot or public transport.While wearing the AR glasses with the Virtual Buzz application and walking home the adolescents have to complete activities proposed by the advisors such as the following:

Interaction with each other: The young people wearing a VR headset with the Virtual Buzz app have to interact with each other for example by shaking hands or giving each other a high five.

Send a message: The adolescent has to write and send a message with their mobile phones to each other.

Traffic sign recognition: From the distance the adolescents try to identify the different traffic signs or to read the street names.

Traffic light recognition: With the red distortion feature of the Virtual Buzz application activated, the adolescents must try to distinguish in the distance the red from the green of the traffic lights.

Catch the ball: The adolescents attempt play with balls of different sizes (f.e. ping-pong, tennis, basket).

Walk straight along a line: An advisor draws a line with chalk on the street on which the adolescents have to walk along without stepping off.

Unlock your home door: Once at home, try to open your door with your key.

10.2.2 Ride a bike

This activity should take place in a closed or delimited space in order to avoid accidents. The adolescents should wear protecting elements like helmets and knee-guards.

Like in a car driving school, the adolescents wearing AR glasses with the Virtual Buzz application should try to complete one or several tours. The tours can increase its difficulty progressively, and the same tour should be traversed with different Virtual Buzz configurations.

10.3 Virtual Buzz extensions with Virtual Objects

These game ideas are targeted at the extension of the Virtual Buzz application using different virtual objects from Unity and the object recognition feature from Vuforia. They could all be implemented in the same app or simply using the features therein as a basis in a new app. Each idea also states whether it would be applicable for single- (SP) and/or multiplayer (MP) modes.

10.3.1 Traffic Sign Recognition

The application shows the user(s) one out of many traffic signs created with virtual objects and a stopwatch is triggered at the same time. The sign could first be shown far away and/or blurred out and then continuously become closer and clearer. Different traffic signs could be associated with different quantities of points. This could for example be done based on the importance of a sign, so identifying a stop sign quickly provides a better score than identifying a speed limit.

If the stopwatch reaches a point where it is too late (sign passed/countdown expired) and the player did not recognize the sign, no points will be awarded and perhaps the drunkenness level lowered if it happens multiple times.

The difficulty could be altered by increasing the velocity, the alcohol levels and or perhaps a reduction of the countdown timer. Correct identifications under a higher difficulty level would then of course be rewarded with more points.

In a single player mode, the user could press a button on the controller as soon as he recognizes the sign. If he/she was correct, points are awarded based on the difficulty and the time it took to recognize the traffic sign.

In a multiplayer mode, the users can compete against each other and are rated either based on their scores or solely on who successfully identified the sign first. The game could be divided into rounds, each one containing for example three recognition sessions. Only the players with at least one positive recognition may advance to the next round. And in the end, the player with the most correct identifications and points wins would be chosen as winner.

Area:	Fun
Pros:	Simple, intuitive game rules Playable regardless of driving experience Simplicity of implementation (SP) No need for a big space/area to play Helps with adjusting to an AR headset
Contras:	Identification done by user should be checked if correct Possibly complex synchronization (MP) Potential loss of the application message against consumption
Impact:	Medium
Effort:	Medium (SP) High (MP)
Technological risk:	Low (Controller not yet used, but simple button presses should not be an issue; traffic signs as blurred/moving virtual objects are possible)

Table 10: Traffic Sign Recognition Evaluation

10.3.2 Avoid Objects

The application creates and throws different virtual objects from different directions with different forms and colors at the user under virtual influence. The user must try to avoid these objects while he receives points for every dodged object and loses points if he was hit. The difficulty can be raised by increasing the user's alcohol level or the object's velocity and movement. The objects could for example not just fly straight at him but slightly change direction or at least appear to due to his drunkenness. The game can be designed not to have rounds but to increases its difficulty as long as the player is able to avoid the incoming objects. This idea would be restricted to a SP mode.

Area:	Fun
Pros:	Simplicity of implementation No need for a big space/area to play Helps with adjusting to an AR headset
Contras:	User has only limited maneuverability May not feel very 'natural' Has hardly any association to driving Potential loss of the application message against consumption
Impact:	Medium
Effort:	Medium
Technological risk:	Low

Table 11: Avoid Objects Evaluation

10.3.3 Object Search

The application briefly displays an object to the user. The user then has to look for the object. If he finds it, he receives points and a new object will be shown. A timer counts down, and the user loses if he does not find the object in the given time.

The difficulty increases based on the points and alcohol level and the given timer may be shortened. The alcohol level increases automatically when the user looks at certain objects and the application recognizes them (using the object recognition feature from Vuforia).

Area:	Fun
Pros:	Simplicity of implementation Helps with adjusting to AR headset
Contras:	The object recognition may give the user an unfair advantage since it is not affected by the alcohol level Needs testing for which objects are sure to work and preparation May feel rather dull Potential loss of the application message against consumption
Impact:	Medium
Effort:	Medium
Technological risk:	Medium (Object recognition feature was not yet tested)

Table 12: Object Search Evaluation

10.4 Virtual Reality Games

This game ideas combine a complete virtual reality world with the image effects used to simulating the influence of alcohol.

10.4.1 Virtual Reality Car Simulator

The application shows a virtual world from the point of view of a car driver under the effects of alcohol. The user can drive by steering a virtual wheel using both of hands and shift up through gears with the controller joystick.

Similar to the physical devices for driving simulation, the driver must drive under different weather situations and avoid different obstacles or challenges. The reaction times are measured and after the session the player can see its reaction times compared to a sober person. The difficulty level could be adjusted automatically as time passes or by using an external controller to influence obstacles, weather situations as well as the level of alcoholic influence.

Area:	Fun
Pros:	No infrastructure needed except for a chair Very appealing to young audience Can be extended in lots of ways SP and MP theoretically possible
Contras:	High production costs Potentially less immersive than physical alternatives due to controls Low chances of being graphically superior
Impact:	High
Effort:	High or even very high
Technological risk:	Very high (complete lack of experience)

Table 13: Virtual Reality Car Simulator Evaluation

10.4.2 Find your way home

Inspired by the last Report from ASN.

The player must go home under the alcohol effects in a virtual world in a given time. The game shows the way home with different virtual object as arrows, pointing fingers, etc.

The player must walk in the right direction, maintain its walking direction, avoid obstacles, go upstairs and downstairs, recognize different traffic signals (f.e. in order to cross the street), avoid running cars, even take the bus/train and get off at the right place.

From time to time the player must complete a kind of task before he/she gets a hint of the way home. With the bluetooth controller, he might need to draw a figure, point exactly towards a checkpoint, or open a door (with the controller acting as the key), etc.

The game is built in different difficulty levels. The alcohol levels and duration of the way increases with each level, or by using a simplified version of the Virtual Buzz Randomization to change the alcohol level in small steps in one direction to simulate getting more drunk or sober.

Area:	User Immersion, Fun
Pros:	Total flexibility in the creation of the environment, challenges or obstacles. SP and MP possible
Contras:	A big physical area without obstacles and preparation is needed Interaction with non-playing persons not possible High production costs
Impact:	High
Effort:	Very High - the creation of a detailed and interactive environment would need many resources and the capabilities of the controller would have to be elaborated first.
Technological risk:	Very high (complete lack of experience)

Table 14: Find your Way home Evaluation

11 Conclusion

We personally are satisfied with how this project has come to pass. We initially were very motivated going into this project, but also quickly overwhelmed with all the information to process. The decision to use the Vuforia SDK for Unity definitely worked out in our favor in the end, but getting accommodated with Unity initially took us a while and we still have much more to learn about all the capabilities as well as the scientific and mathematical processes behind them.

The lack of concrete or SMART goals at the start of the project also came with quite a bit of anxiety and project management issues regarding what to prioritize, but we managed that alright and ended up fulfilling most of the goals identified by both our industrial partner and ourselves in cooperation with Prof. Augenstein. Further into the development process, we also ended up losing a lot of time due to minor malfunctions or bugs with no obvious origin that we could have used for the final finish, but we remain happy with the results.

By successfully developing a first, working prototype of an app that mostly satisfies the initial requirements and can already be used as it is by our industrial partner, we have reached the best-case scenario for the initially uncertain idea. This also meant, that we had some time to allocate to the perfection of the small codebase used for the app as well as of course the documentation. The information we have provided in the form of our little developer blog, this document as well as the comments within the code should easily suffice to give any interested developer (or even non-developer) a sneak peek into the technologies used in this project and what can be expected in terms of AR. Considering the still very frequent steps forward in new technologies in this area and the very limited subset of features we have actually used, there could have been more and better results especially regarding the provided game ideas and feature outlook, but we think it is a good start.

We are really grateful for the opportunity to use this research oriented project to gather information and first experiences in this interesting area of development as well as the pleasant cooperation with both our industrial partner with all their feedback and the scientific guidance by our advisor, Prof. Augenstein.

The perhaps most important conclusion we could infer from this project is, that although the AR/VR technology is mostly ready and very capable, the available hardware resources (at least regarding smartphones) are still struggling with providing a fluent and comfortable experience. In terms of computer based VR, the experience is much more fluent due to the much more powerful components, but still comes at a harsh tradeoff in graphics quality. It feels safe to say though, that the hype of technologies in these areas will most definitely continue to go on and we are ecstatic to see where this will lead us.

12 Table of Figures

13 Images

14 References

[1] "Drogisto: Alkoholbrille," [Online]. Available: https://www.drogisto.de/produkte/details/alkohol-3/rauschbrillen/alkoholbrille.html. [Accessed 21 12 2016].

[2] P. Milgram and T. Haruo, 1994. [Online]. Available: http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf.

[3] 11 10 2016. [Online]. Available: http://www.businesswire.com/news/home/20161101006531/en/PTC-Unity-Announce-Strategic-Collaboration-Accelerate-Augmented.

[4] Unity Technologies, 2016. [Online]. Available: https://unity3d.com/profiles/unity3d/themes/unity/images/company/pr/unity-vision-mobile-graph.jpg.

[5] Unity Technologies, 2016. [Online]. Available: https://madewith.unity.com/.

[6] 12 9 2014. [Online]. Available: https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/.

[7] Unity Technologies, "Unity Manual: Blur (optimized)," [Online]. Available: https://docs.unity3d.com/Manual/script-BlurOptimized.html.

[8] R. Fisher, S. Perkins, A. Walker and E. Wolfart, "Image Processing Learning Resources: Gaussian Smoothing," [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. [Accessed 25 12 2016].

[9] Unity Technologies, "Unity Manual: Motion Blur," [Online]. Available: https://docs.unity3d.com/Manual/script-MotionBlur.html. [Accessed 25 12 2016].

[10] Unity Technologies, "Unity Manual: Camera Motion Blur," [Online]. Available: https://docs.unity3d.com/Manual/script-CameraMotionBlur.html. [Accessed 25 12 2016].

[11] "UAH Math Resources: The Simple Random Walk," [Online]. Available: http://www.math.uah.edu/stat/bernoulli/Walk.html. [Accessed 23 12 2016].

[12] "UAH Math Resources: Standard Brownian Motion," [Online]. Available: http://www.math.uah.edu/stat/brown/Standard.html. [Accessed 23 12 2016].

[13] Unity Technologies, "Unity Manual: Color Correction Curves," [Online]. Available: https://docs.unity3d.com/Manual/script-ColorCorrectionCurves.html. [Accessed 23 12 2015].

[14] C. Dickinson, Unity 5 Game Optimization, Packt Publisher, 2015.

[15] Qualcomm Vuforia, 14 12 2016. [Online]. Available: https://www.vuforia.com/-/media/Vuforia/Homepage/Singles/Vuforia%20Logo%20OLx2.png.

[16] [Online]. Available: http://www.fachstelle-asn.ch/images/asn-logo-de.png.

[17] G. Abramovich, "CMO: ADI Gaming Report," 16 9 2016. [Online]. Available: http://www.cmo.com/adobe-digital-insights/articles/2016/8/11/adi-us-gaming-report.html#gs.BvbHzyU.

15 User Manual

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

Virtual Buzz:

Simulating visual Influences of Alcohol in an Augmented-Reality App

User Manual

16 Table of Contents

1 Installation

1.1 APK File

An APK file is the file format used for installing software on the Android operating system. This is the file used to install the Simulating Substances application. Since this app is currently not to be found on any app store and the decision to do that is not planned as of yet, this is expected to stay the primary way

1.1.1 Activate Developer mode

As there are different android devices, it’s recommended to search in Internet how to activate this service on your device.

Generally you must navigate in your phone to the “Build number” portion of the settings and tap on the section 7 times.

After two taps, a small pop up notification should appear saying "you are now X steps away from being a developer" with a number that counts down with every additional tap. After the 7th tap, the Developer options will be unlocked and available.

Please keep in mind, that this information may be different on your device.

1.1.2 Allow third-party apps

Before you can install the application on your phone you will need to make sure that third-party apps are allowed on your device since the installation-file does not come directly from an app store and therefore is not signed.

Go to Menu > Settings > Security > and check Unknown Sources to allow your phone to install apps from sources other than the Google Play Store. Please keep in mind, that this information may be different on your device.

1.2 Installation

After obtaining the “SimulatingSubstances.apk”-file, you just have to find it on your phone, tap it, and then hit Install. And you're done.

2 Introduction

2.1 Scenes

The term 'Scene' referenced throughout this tutorial refers to the individual screens (as a composition of different UI-elements) accessible within the app and has its roots in the Unity game engine we used to create the app. This chapter will provide a quick explanation of the scenes in our app.

2.1.1 Main Scene

This scene is the entry point of the app and gives you the option to navigate directly to all of the other scenes using the listed options.

2.1.2 Simulating Scene

This is where the 'action happens' and the visual effects used to simulate the influence of alcohol are applied based on the selected configuration. While in this scene you are expected to wear a device such as Google Cardboard.

The effects in play are displayed on the upper part of the screen and the buttons used to navigate away from this scene are located on the bottom.

2.1.3 Configuration Scene

This scene is used to create your own configurations as a set of settings for each of the simulated effects available with an option to permanently save the current selection on your smartphone under a chosen name.

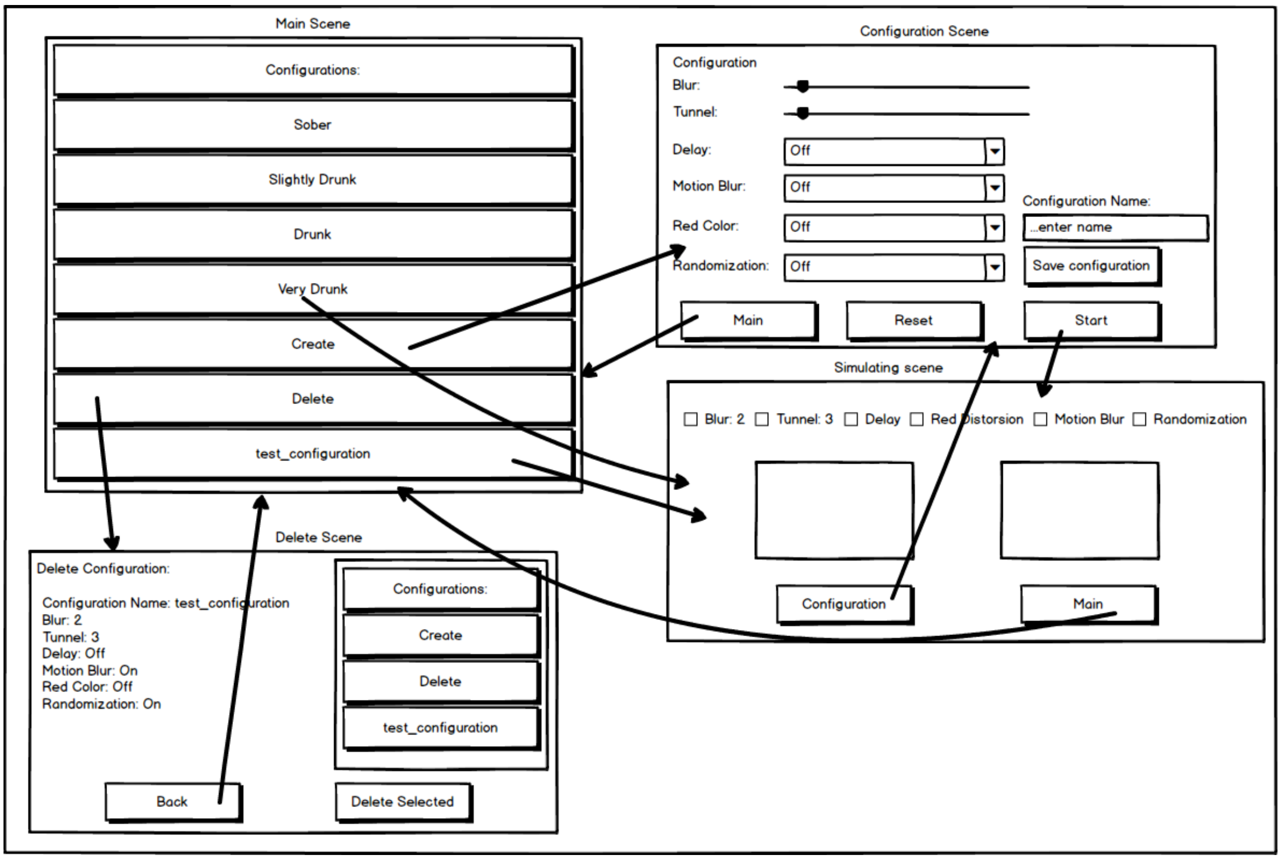
Any configurations saved will then appear in the listed options in the "Main Scene" and be available for selection until you either delete the file using the "Delete Scene" or do so manually. Since the configurations are saved in files on your smartphone, you could also transfer them onto another device manually.

2.1.4 Delete Scene

As the name indicates, this scene is used for the deletion of custom configurations as an alternative to manually deleting the files created in the app's directory on the phone. To make sure you delete the right configurations, an overview of the settings stored within the selected configuration is provided on the left and in case any error was encountered during the deletion-process, a message will appear in the same place accordingly.

2.2 Graphical User Interface Overview

The image below provides a better understanding of the navigation in the Simulating Substances application. The entry point to the application is the Main scene:



Delay: Off
Motion Blur: Off
Red Color: Off
Randomization: Off

Configuration Name: test_configuration
Blur: 2
Tunnel: 3
Delay: Off
Motion Blur: On
Red Color: Off
Randomization: On

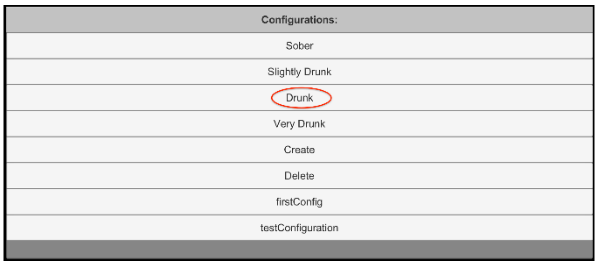
3 Use Cases

3.1 Start with a default configuration

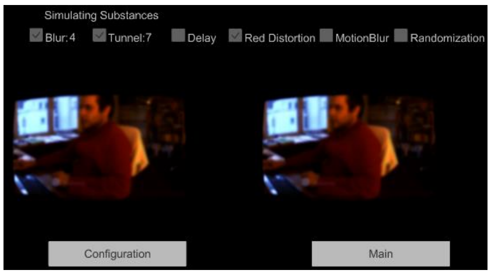
The application comes with the following four default configurations, which are supposed to resemble different stages regarding the influence of alcohol:

1. Sober
2. Slightly Drunk
3. Drunk
4. Very Drunk.

After starting the application, the "Main Scene" is loaded. There you have the option to select a default configuration from the list by **double-clicking** it. The selected configuration will then be loaded and used by the "Simulating Scene" that is started therewith and you can put your phone in a fitting device such as Google Cardboard.

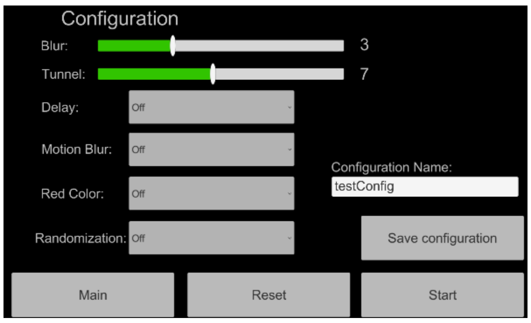


→



Blur:4 ✓ Tunnel:7 ✓ Delay Red Distortion ✓ MotionBlur Randomization

3.2 Create a custom configuration



Blur: 3
Tunnel: 7
Delay: Off
Motion Blur: Off
Red Color: Off
Randomization: Off
Configuration Name: testConfig

In the "Main Scene" **double-click** on the "**Create**"-option.

The Configuration scene will then load and allow you to create a custom configuration with any combination of the displayed options.

Choose the Blur and Tunnel View values by sliding the slider to the left or to the right. Activate or deactivate the Delay, Motion Blur, Red Color Distortion or Randomization effects just by selecting "On" or "Off" in the corresponding dropdowns.

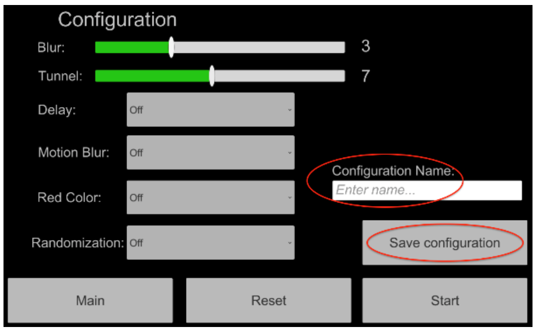
Once finished, hit the Start button and the simulation will start.

If needed it's possible to reset all the values to 0 or "Off" by pressing the Reset button.

3.3 Saving a custom configuration

Follow the steps described in “Create a custom configuration”. Before starting the simulation, enter your chosen name for the configuration in the field and hit the “Save configuration”-button.

A message will appear once the saving process has been completed.



Configuration

Blur: 3

Tunnel: 7

Delay: Off

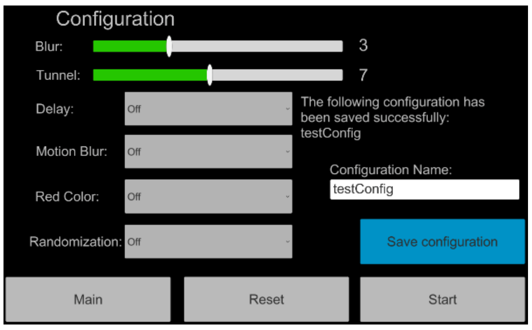
Motion Blur: Off

Red Color: Off

Randomization: Off

Configuration Name: *Enter name...*

→



Configuration

Blur: 3

Tunnel: 7

Delay: Off

Motion Blur: Off

Red Color: Off

Randomization: Off

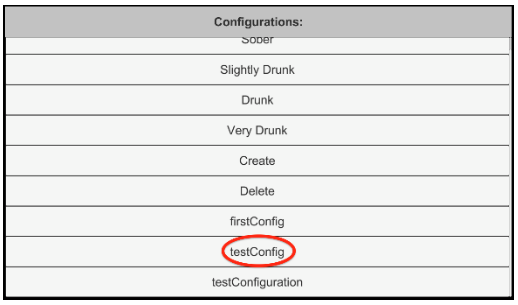
The following configuration has been saved successfully: testConfig

Configuration Name: testConfig

Hit the “Start”-button in order to begin the simulation and test the configuration or use the saved configuration by selecting it on the list in the “Main Scene”.

3.4 Starting with a custom configuration

Similar to “Start from a default configuration”. In the Main scene scroll the list down until you find the desired custom configuration. **Double-click** the item and the simulation will start.



Configurations:

Sober

Slightly Drunk

Drunk

Very Drunk

Create

Delete

firstConfig

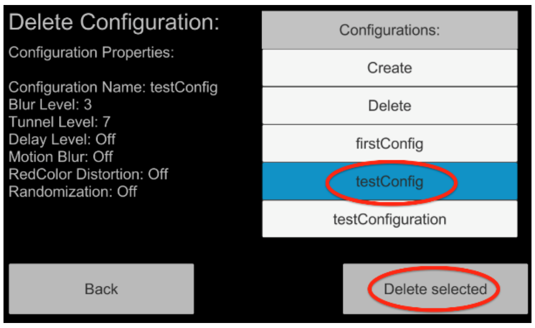
testConfig

testConfiguration

3.5 Delete a custom configuration

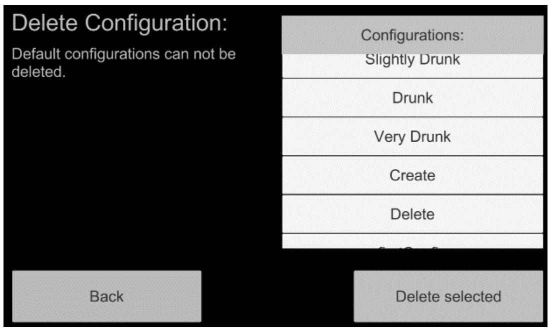
In the “Main Scene” select **Delete** from the list and **double-click** on it. The “Delete Scene” will be loaded. In the Delete scene you can check the values of the different configurations by tapping on the list.

If you want to delete a configuration, **double-click** on it in the list and then hit the “Delete selected”-button. If you choose to delete a default configuration an error message will appear.



Delete Configuration:
Configuration Properties:

Configuration Name: testConfig
Blur Level: 3
Tunnel Level: 7
Delay Level: Off
Motion Blur: Off
RedColor Distortion: Off
Randomization: Off



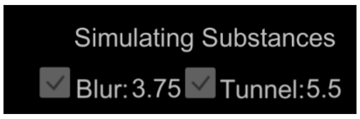
Delete Configuration:
Default configurations can not be deleted.

3.6 Activating the Randomization Effect

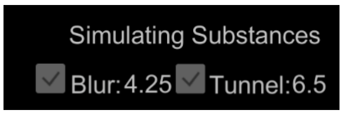
The randomization effect is a feature, which tries to imitate the inconsistency of the alcohol effects by changing the Blur and Tunnel view effects strength on the runtime after a time interval of five seconds.

This effect is conceived to increment or decrement this values in a progressive way, avoiding big steps which could be very disturbing for the user and has a probability of 30% to remain the same.

After activating this effect in the "Configuration Scene" you can see any changes applied to the values in runtime in the upper corner of the "Simulating Scene":



→


Simulating Substances
✓ Blur:4.25 ✓ Tunnel:6.5