TU WIEN Informatics

Cardiac blood flow visualization in marker-based augmented reality for educational purposes

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Miran Jank
Registration Number 01526438

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dr. Renata Raidou

Vienna, 24th October, 2022

Miran Jank	Renata Raidou

Erklärung zur Verfassung der Arbeit

Miran Jank

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 24. Oktober 2022

Miran Jank

Danksagung

Zuallererst bedanke mich herzlich bei meiner Betreuerin Assistant Prof. Dr.Renata Raidou, für die Gelegenheit mit ihr an diesem großartigem Projekt arbeiten zu dürfen. Ihre ausgezeichnete Hilfestellung, durchgehender Support und tiefgreifende Erfahrung, ohne welche diese These nicht möglich gewesen wäre, hat mich bis zum Abschluss dieser Arbeit stehts begleitet und unterstützt.

Weiters möchte ich mich bei meiner Familie, bestehend aus Software-Entwicklern (und daher auch Testkandidaten), für die geleistete technische, moralische und emotionale Unterstützung bedanken.

Nicht zuletzt, danke ich hiermit allen Studienprobanden und -probandinnen, ohne welche diese Studie nicht möglich gewesen wäre.

Acknowledgements

Firstly I would like to thank my supervisor Assistant Prof. Dr. Renata Raidou for the unique opportunity to work with her on this fascinating project. Her excellent assistance, continuous support and profound experience, without which this thesis would not have been possible, has constantly accompanied and supported me until the completion of this work.

Furthermore, I would like to thank my family, consisting of software developers (and therefore also test candidates), for their technical, moral and emotional support.

Last but not least, I would like to thank all study participants, without whom this study would not have been possible.

Kurzfassung

In den letzten Jahren hat sich der Stand der modernen Bildung drastisch verbessert, jedoch fehlt es im Bereich der medizinischen Ausbildung noch immer an Innovation. Mit dieser empirischen These wollen wir medizinische Lernansätze für junge Studenten modernisieren, indem wir eine immersive, einprägsame und ansprechende Visualisierung von Inhalten entwickeln. Trotz ihrer vielfältigen Vorteile, werden visuelle Modelle, welche ein großes Potenzial zur Steigerung des Lerneffekts haben, leider nur in begrenztem Umfang in der Ausbildung eingesetzt.

Mit dieser These schlagen wir eine Methode zur Verbesserung des modernen Anatomie-Unterrichts vor, unterstützt durch die von uns entwickelte, Marker-basierte Augmented Reality (AR) Anwendung *CARdiFlow*, welche auf jedem gängigen Smartphone verwendet werden können soll. Unser Ansatz erleichtert es den Nutzern, das kardiovaskuläre System sowie die Struktur und Funktion des Herzens im Detail zu verstehen. Durch unsere neuartige Herangehensweise an dieses Problem haben wir auch einen Weg gefunden, realistische Flüssigkeiten dynamisch auf mobilen Geräten darzustellen, die mit modernster Technologie ausgestattet sind, und gleichzeitig AR-Inhalte zu rendern.

Wir untersuchten außerdem komparativ die Fähigkeit der Studenten, Herzblutströme und die Herzstruktur in Abbildungen zu erkennen, nachdem sie entweder mit traditionellem Lernmaterial, d. h. statischen Referenzbildern und beschreibendem Text, oder mit unserem modernen Ansatz, der AR-Anwendung *CARdiFlow*, konfrontiert wurden. Die verschiedene Testpersonen (Kontrollgruppe `n = 13`, Experimentalgruppe `n = 13`) erhielten ihre Instruktionen via Portable Document Format (`.pdf`) und ihre Prüfungen wurden ausgewertet. Zusätzlich waren wir an einem Vergleich der Benutzerfreundlichkeit der AR-Anwendung im Vergleich zu den traditionellen Lernmitteln interessiert. Daher setzten wir einen System Usablility Scale (SUS) Test ein, um noch mehr Feedback abzuleiten.

Die Studenten, welche die AR-App benutzten, zeigten ein besseres Verständnis des kardialen Blutflusses und schnitten bei den Aufgaben zur Herzstruktur signifikant besser ab. Außerdem stellten wir fest, dass die Nutzer länger mit den Lernmaterialien arbeiteten, was auf ein besseres Engagement aufgrund intensiverer Immersion hindeutet. Dies wurde auch durch die SUS-Ergebnisse bestätigt, welche die Benutzerfreundlichkeit unserer Anwendung mit der Note *A* bewerten, verglichen mit der Note *D* für traditionelle, statisch-illustrative Darstellungen.

Abstract

In recent years the state of modern education has improved drastically, but innovation is still lacking in the department of medical education. With this empirical thesis, we aim to modernize medical learning approaches for young students by introducing immersive, memorable and engaging content visualization. The cardiac blood flow is a complex process to grasp for laymen, especially if it comes to the various mechanical in- and outflows from and to the human heart. Despite their benefits, visual models, which have great potential to increase learning effects, are still only used in limited educational settings.

With this work we propose a method to enhance modern anatomical teaching, supported by marker-based augmented reality (AR) on any smartphone, through the application we developed, *CARdiFlow*. Our approach facilitates users to understand the cardiovascular system, as well as the heart structure and function in greater detail. Through our novel approach to this problem we also found a way to display realistic fluids dynamically on mobile devices equipped with state of the art technology, while still rendering AR content.

We additionally investigated comparatively the ability of students to label cardiac blood flows and heart structure in figures after being exposed to either traditional learning material, being static reference images and descriptive text, or our modern approach with the AR application *CARdiFlow*. Various test subjects (control group `n = 13`, experimental group `n = 13`) were given their instructions via Portable Document Format (`.pdf`) and their exams were evaluated. Additionally, we were interested in the comparison of the AR application usability compared to the traditional means of learning, and therefore employed a System Usablility Scale (SUS) test, to deduct even more feedback.

The students who experienced the AR activities showed a increase in the understanding of cardiac blood flow and scored significantly better on heart structure-related tasks. We also noticed an increase in the time users handled the learning materials, therefore indicating longer engagement due to the level of immersion. This was also confirmed by our SUS results, which place the usability of our application at grade A, compared to grade D of the traditional (i.e., static illustrative) representations.

Contents

CHAPTER 1

Introduction

This chapter gives a quick introduction to the state of education, which we try to enhance with this thesis, as well as an overview of the motivation behind our work. Subsequently we provide a short introduction to most important aspects covered by this project. Further, related work is discussed in detail.

1.1 Motivation

Anatomical education is a big part of contemporary educational plans, starting from middle school biology and extending to many different study programs. Sadly, most of the conventional means are limited to displaying information in a static way, meaning representations without movement, interaction, or details-on-demand. Simply put, there are no dynamics to entice students into learning.

The goal of this thesis is to enrich the process of learning about anatomy. We choose the cardiac blood flow as a specific domain of application. In this field, there is a lot of potential for flow visualization, which is not presented well in traditional, static depictions. By augmenting the blood flow in a static 3D model of the human heart, we may be able to focus much of the processing power of an augmented reality (AR) device towards educational visualizations. This enables the user to interact with the model, and potentially to understand the interplay of ventricles and atrium, flow directions and architecture of the heart easier than with, e.g., conventional static anatomical illustrations.

How can we design a free standalone augmented reality application, which visualizes cardiac blood flow for educational purposes? This question has not been yet comprehensively and comprehensibly answered. As there is no available software on the free market, this thesis aims to provide a solid base for further research.

Observing the market for AR apps, we can quickly identify that the biggest platforms for distributing such applications are those, which cater to handheld devices like smartphones [Cra13]. Alternatively there are also better developed systems like Google Glass [Gooa] and Microsoft Hololens 2 [Mic]. These will not be considered to be part of the target market, as they specifically cater to professionals. For our target audience, there are two major competitors, which dominate the market: the Google Play Store [Goob], as well as the Apple App Store [App], according to research by Ali et al. [AJM17]. Scanning these platforms has shown that there is a clear lack of free software concerning education in AR, as a whole. Further research confirmed that there is only one paid competitor, namely Virtuali-Tee by Curiscope [Cur], which focuses on displaying the inner human bodies by marker-based AR. Although their app is free, it only works with their specialized T-shirts, which need to be bought and delivered. Most inportantly, Virtuali-Tee is also not supporting dynamic blood flow visualization. Our approach focuses on providing a self-sufficient solution that supports an engaging cardiac blood flow visualization.

1.2 Cardiac Blood Flow

The main focus of this thesis is the visualization of blood flow in the human heart. This is no easy task though, as traditional ways of visualizing flows are inanimate and static — mostly, raw drawings of fluids in 2D space. Sometimes these illustrations are enhanced with different approaches, e.g., flow arrows in hedgehog plots, streamlines, streamribbons or -surfaces, glyhps and icons. Seldom, there are free 3D visualizations available via media-sharing websites on the world wide web, which offer more sophisticated representations, like (in)direct visualization, or time dependent simulations. This approach of a realistic time-dependent simulation of the cardiac blood flow will be used in *CARdiFlow*, the AR application result produced in this thesis.

1.3 Augmented Reality

Augmented Reality (AR) is a modern technique to visualize real world elements in conjunction with real-time computer generated data. It is found to be close to reality, especially in the space of the well defined Virtuality Continuum (see Fig. 1.1) [MTUK94]. AR is part of the expanding Extended Reality (XR) umbrella term, encapsulating well-known terms like Virtual Reality (VR), which excludes real world elements and bases solely on generated data, or Mixed Reality (MR), which enables simultaneous interaction with real world elements and generated data [MTUK94].

Augmented Reality is mostly used, if interaction and enhanced user engagement is the center point of the project. It enables users to interact with the model, meaning zooming, panning and labeling among many other technologies facilitating user entertainment. We chose to include all of these practices, as it should facilitate user engagement as well

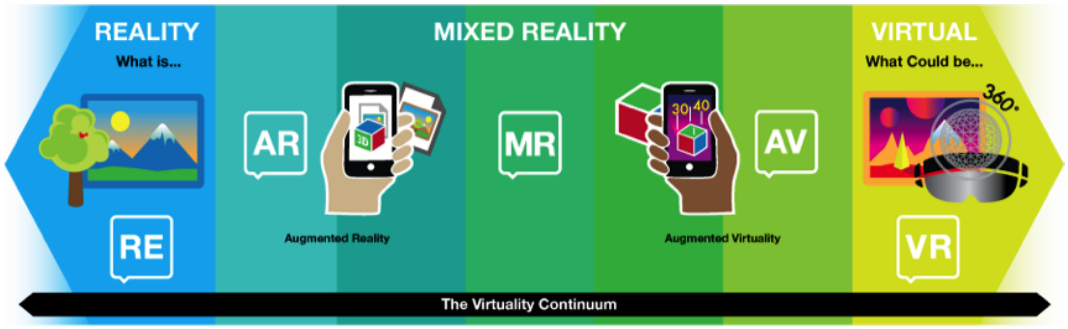


Figure 1.1: A rough depiction of the Virtuality Continuum by Milgram [MTUK94] et al. Image by Anderson [And]

as the motivation of users to learn. It is of utter importance to engage the student, as "previous research has identified the problem that technology will create a passive learning process if the technology used does not promote critical thinking, meaning-making or meta-cognition." [SAhY15]. With the implementation of advanced mechanisms we are on a promising path, or as Saidin et al. state: "Since its introduction, augmented reality (AR) has been shown to have good potential in making the learning process more active, effective and meaningful." [SAhY15].

1.4 Target Audience

This thesis' outcome aims to enhance the learning process of cardiac blood flows, especially in younger generations. We think that the added benefit of interaction may provide an increased amount of user motivation and understanding of stale study matter, as was proven multiple times by different theses using this same approach. Akcayir et al. state in their paper that there are many advantages as well as challenges associated with augmented reality for education [AA17]. They highlight that "[...] while some studies reported that AR decreases cognitive load, others reported that it causes cognitive overload". They link this occurrence to the non-existent technological know-how of the educators, as well as the youth of AR in general. Though they also state "with expected technological developments, the utility of AR technology should improve in the near future. Thus, its usage likely will be more widespread, and more research will be focused on it.". This being said in 2016 also validates our assumptions. Another work by Zhu et al. provides a more narrow and detailed sight on augmented reality in healthcare education [ZHMZ14]. They mention that "AR is in the early stages of application within healthcare education but it has enormous potential for promoting learning in healthcare based on this review of preliminary AR studies.". This further motivates the implementation of augmented reality in the context of healthcare.

As our focus group we see children, teenagers, as well as young adults in a range from 10 to 20 years. Their inherent knowledge with nowadays technology as well as strong motivation to learn and understand are a perfect fit for AR solutions. Though we do not limit ourselves to this range of age, as we truly believe that *CARdiFlow* is able to enable any layperson to learn cardiac blood flow in a simpler, more native way.

People we can not account for in our approach are students and professionals in the various fields of medicine, as they are already very knowledgeable, and thus will probably not see any profit in using our product. It may still motivate them to reflect on their already well-developed understanding of the heart, but they will not see any improvements to their medical education, since we also do not plan to design our software for advanced blood flow education.

1.5 Related Work

As the field of flow visualization is well researched, we tried to build on the established standards and incorporate the best practices achieved by Post and Walsum [PW91]. As their book covers a wide range of flow visualizations, it is a perfect work of reference for everything flowing, be it smoke, fluid or other. According to them, "experimental flow visualization techniques are applied for several reasons:

- to get an impression of fluid flow around a scale model of a real object, without any calculations;
- as a source of inspiration for the development of new and better theories of fluid flow;
- to verify a new theory or model. [PW91]"

We clearly see our project in the first category, making sure, that students understand the cardiac blood flow better than with stale, old, traditional means of education, e.g., paper sketches and verbal communication.

A thorough research of the field has shown that similar applications to ours do not exist yet. Though there are plenty of AR apps, which try do educate their viewers on bodily functions and anatomy, the closest competitor is Virtuali-Tee [Cur]. Their approach does not visualize blood flow, as they only display the outer bounds of every organ, leaving a lot of space for imagination. To really understand the cardiac blood flow, it is essential to see the dynamics of the blood, in motion. We will enable this, and underline it with different visualization and interaction techniques, such as zooming and labeling.

A recent study by Gonzales et al. has shown that AR applications have a significant impact on students, who try to learn cardiac physiology [GLP+20]. This shows that

it is possible to achieve a learning performance boost for students, so we try to elaborate on this and extend it to even laypersons and younger generations.

As blood flow measurement and visualization has already been an important factor in the past, we will try to incorporate as much related research for our contribution as possible. Preim et al. mentions that computational fluid dynamics (CFD) aid in blood flow simulation [PB13], which we found to be very useful, since it is reducing the overall computing cost for our AR-application, which will need to run on hand-held devices. For creation of this CFD animation we will use the widely known 3D modeling software `Blender` [Ble]. Files will be stored locally via the Alembic (`.abc`) standard [Son], providing us with a compact file, which encompasses all meshes.

Blood flow needs to be measured quite in detail, meaning displaying it will also incur a lot of performance issues. As Köhler et al. state "These unsteady vector fields on the complex domain of the human aorta are difficult to visualize effectively, in particular in case of pathologies and the resulting complex flow behavior." [KBvP+17]. It is of big importance, to make the simulation performant and designing the visualization effectively. Our approach builds on the real-time graphics engine `Unity` [Tecc] and the underlying `C#` programming language, trying to get the best performance out of it.

We would also like to stay as true as possible to the properties of real-life blood to increase the amount of realism. Therefore, we based our blood consistency and color values on various reliable sources, such as the work of de Hoon [dH13, dHvPJV14]. `Blender` as a simulation platform enables us to define a multitude of variables, all being used in the computational fluid dynamics simulation. Also the material variables are set by `Blender`, giving us full control of the blood color, roughness, highlights and so forth. By streamlining the cardiac blood flow to be fully animated in one software suite, we hope to get better consistency and enable future automation.

AR in combination with educational anatomy has been well researched over the past years. It has been proven that although the learning efficiency with AR is as potent as with digital media like tablets, student engagement and willingness to learn are significantly higher [MSRS17]. We will implement the AR aspect via the `Vuforia` API [PTC], one of the bigger AR engines currently on the market [SSJ16]. This way we will get good, precise marker-based tracking, to ensure natural handling for end-users. Furthermore, through labeling and interaction, as already widely used by others, we try to engage their attention.

Better interaction enhances the learning process — an excellent example showing this is the work of Saalfeld et al. [SOJS+17]. They show that with the mere addition of sketches and annotations, students could be helped during their education in the medical domain. Furthermore, even treatment planing and patient education may benefit in a significant way from the addition of more interaction possibilities. We plan to annotate as much as needed, while reducing visual overhead as much as possible.

CHAPTER 2

Technical Background

2.1 Data Set

The main contribution of this thesis is an intuitive application for the visualization of the human heart model, as well as the blood flowing through it. In order to stay as close as possible to reality, we need to incorporate the most realistic data.

2.1.1 Blood

Blood is an opaque, red fluid in humans, which is denser and more viscous than water. It has an index of refraction (IOR) of around $\sim 1,75$ [Elb21], depending on the amount of hemoglobin in it, which also darkens or brightens it when (de-)saturated with oxygen. This way our blood can range from crimson to dark-brown colors, most often being represented by ISCC-NBS descriptors as "deep reddish brown" to "vivid red". This can be translated onto our screens as Red Green Blue (RGB) color model values ranging from (RGB = 97, 15, 18) to (RGB = 213, 28, 60). For this work we decided on a value of (RGB = 128, 13, 0) with a transparency of (α = 192). Its surface tension is ~ 55.9 millinewtons per meter ($mN \cdot m^{-1}$) [EH97], making it less tense than water at ~ 72.9 ($mN \cdot m^{-1}$), and therefore less likely to shrink to the minimal surface area possible. Blood represents roughly $\sim 7\%$ of the human body weight, in average adult humans this means a volume of around 5 liters correlating to $\approx$60 milliliters per 1 kilogram of body weight. Its function is to carry nutrients and oxygen to every cell in the body. But simultaneously, blood transports wastes away from cells. This process is controlled by the cardiovascular system that consists of the heart, vasculature network and the blood that blood vessels transport. In an average adult human blood flows with a systolic (blood during ventricular contraction) pressure of 120 mm Hg and diastolic (blood during ventricular relaxation) pressure of 80 mm Hg. Though values vary extremely with different blood vessels, as can be seen in Fig. 2.1.

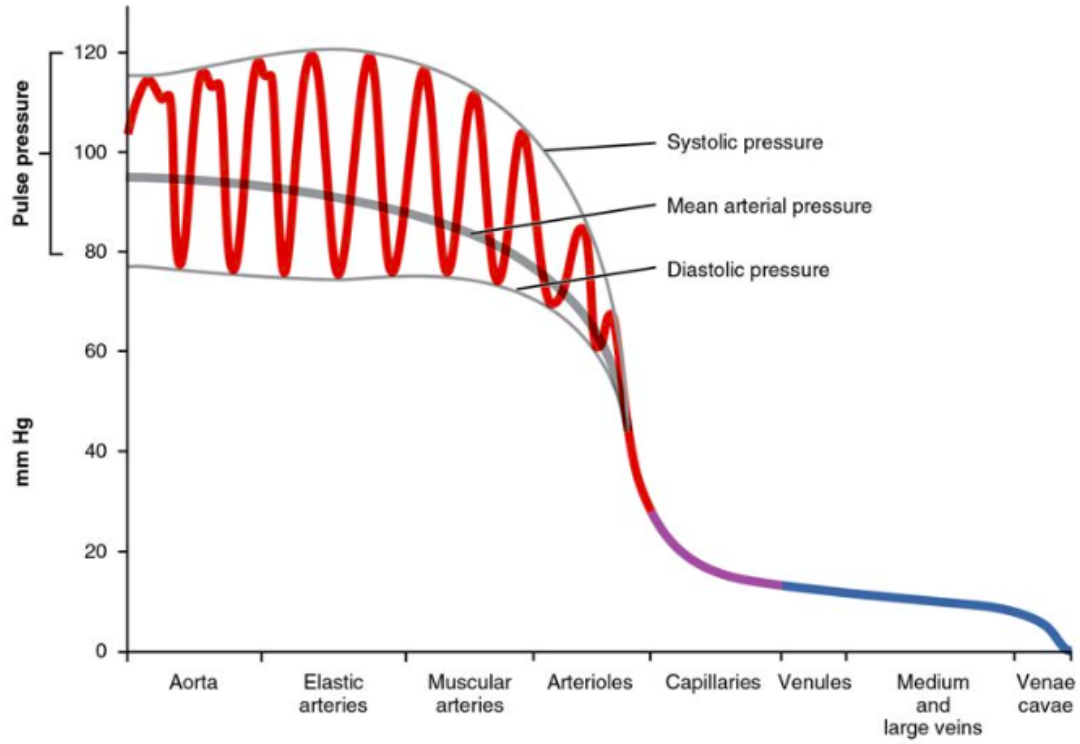


Figure 2.1: Blood pressure is higher in vessels with high diameter, and falls off in lower ones [Lum].

Since the cardiac blood flow will be pre-generated, it is of utter importance to be as precise with its simulation as possible. Blood is, although similar to water in some criteria, quite different to most liquids, and thus has to be calculated very differently. For simulation and visualization there are many factors, e.g., viscosity for simulation and the index of refraction (IOR) for visualization among many others, which we based on the research of de Hoon [dH13, dHvPJV14]. For the calculation and file generation we use the popular free 3D modeling software `Blender`, which enables us to pre-calculate the flow visualization and export it in a way, which can then be imported into `Unity`.

2.1.2 Heart Model

The realistic model of the human heart [CGZ] was obtained from the popular 3D model sharing website turbosquid.com [Tur]. It was chosen because of the models high level of detail, while still maintaining reduced complexity by having small triangle counts. Another factor being already modeled open, so viewers can instantly observe the blood flowing. A free alternative [Rai] has proven to be unreliable, due to having multiple holes to fill and inconsistent or completely non-realistic proportions. However, there are

also many models, which are not textured and therefore can not be accounted for, as providing this by ourselves would massively exceed the projects time limitations.

2.1.3 Limitations

Another interesting feature would be to animate the heart, and thus get physically correct flows according to pressure levels. This would, although quite important, explode in complexity and is thereby not implemented in this thesis. However, we see this as a good starting point for future work, as it could further enhance the students' understanding of the workings of the heart.

2.2 Unity and C#

A requirement we set is, that the result should be scalable and easily modifiable, or extensible to other solutions. This therefore excluded some of the system-near applications like Android Studio [LLCa] and its supported programming languages `Java`, `Kotlin` and `C++`, as they would not allow for the level of abstraction needed for this task. Competitors of interest were the well-known Unreal Engine [EG] as well as A-Frame [AFr], as they also enable quick prototyping and already include valuable libraries that support their production-ready environment.

Among all these the `Unity` Engine[Tecc] stood out as a prime example of simplicity, level of abstraction and already contained functionality, building on the `C#` programming language — known for its fast execution times although being completely object-oriented. With this tool-set it was easy to generate executable programs for all main platforms, including Windows and Android. `Unity` even enables users to quickly run their application in editor mode, before building the project, to allow for even quicker feedback iteration.

2.3 Vuforia API

`Unity` by itself provides many optional libraries for AR development like, e.g., `Unity` Mars [Tecd], `Unity`'s AR Foundation Framework [Teca] or the XR Interaction Toolkit [Tece] which also includes VR functionality, but at the core all these libraries are just an interface to even deeper laying API's, for example ARCore [LLCb] and ARKit [Inca].

`Unity` also allows one to embed external work-tools, as we did with the renown `Vuforia` API [PTC]. Its scalable and comprehensive code base are a perfect fit for this project and allow us to use a variety of features while staying consistent and performant. Since they are one of the industry leaders concerning AR, their technology allows for tackling even the most advanced problems with ease, enabling us to stay free while prototyping and not having to be limited by narrow possibility perimeter. `Vuforia` offers a free student account with limited access to their features, but also gives a small amount of

their online cloud capacity in return. This let us quickly evaluate our image targets and convert them into a format readable for Unity.

2.4 Markers & Image Targets

We also need to visualize the heart model for the users and to keep that model trackable by the Vuforia API. In order to achieve this, usually the method of marker-based tracking is used. A marker is simply a physical object, with as many diverse, but discernible features, as possible. It is used, to facilitate the processing of image data in the API. Without it, it would be impossible for Vuforia to know, where to place the generated 3D model.

Vuforia's API calls their markers image targets, as of the analogue thought of an printed out image, which is therefore our target for model visualization. In this thesis we will use the term marker for the real world counterpart, and the term image target for the in-code representation. It is also be possible to define other structures than just planes as image targets, such as cuboids, cylinders and even custom predefined 3D objects.

Thus, it is of utter importance to design image targets in such fashion, that their features are best readable by machines. This means many little, but precise features, which would not make much sense for the average human, are a perfect fit. Though in-practice there is often a little representation, or at least an abstract version of the in-engine generated 3D model, depicted right in the center of the marker. This helps the user to understand what should be shown on their device, and in case of multiple marker tracking to differentiate between them.

2.4.1 Physical Properties

The target material for a marker should be hard and not flexible, as it is crucial for the perspective part of texture to be as constant as possible. For example, if an object were to be folded exactly at the middle part, only half of the needed features would be visible, and therefore would not be trackable. This of course is a matter of tracking definition, as we can set it in code to also follow objects, which do not fit our required features 100%. This is not recommended though, as it may detect other, unknown objects in the scene and falsely use them as image target.

Another important point of view is that most cameras nowadays are equipped with auto-focus. A bendable material, as for example paper, could therefore get out of focus and also hinder the recognition process. This is why many AR-projects tend to give out hard samples, or at least inform the user to support the paper by adding non-flexible pasteboard to the backside of the marker. Another easy way to circumvent this, would be to instruct users to use thicker print paper, with a density of $\sim$ 200 to 220 g/m^2.

Another key factor for best possible machine recognition is the marker's size. Usually a rule of thumb is at least 12 cm in width, and a reasonable height. However, size really depends on the camera's distance to the viewed marker. Vuforia recommends at least 10 cm in width for each Meter of distance, based on empirical observations [PTC]. This is to be taken with caution though, as different cameras with different resolution, point precision, light intake, focal length, pre- and post-processing software, etc. can interfere with this measurement.

The surface of every marker should be flat, but rough. A very smooth surface, for example metals or photo paper printed on by a laser printer, may reflect a lot of sun hindering the vision of cameras. It is also recommended to be in a lightly lit place, where there is no direct sunlight, to avoid these artifacts.

2.4.2 Best Practices

There exist a range of key attributes, useful to designing well thought out image targets [Vufa]. Among them we determined four with great impact: detail richness, good contrast, no repetitive patterns and file format. Concerning *detail richness*, it would be best to have a image with a very heterogeneous image histogram, as can be seen in Fig. 2.3. This provides a higher amount of features, which will be discussed soon. *Good contrast* is always a good spot for machine recognition, as areas with bigger differences in color values are more easily distinguishable. It is even possible to artificially improve the contrast by using local contrast enhancement, more on this later. *Repetitive patterns* might at first seem like a good approach to designing image targets, though this is easily debunked, if we think for example about a chess board. There are many features, and the contrast is also well settled, but there is no way to tell where up or down would be, neither right or left. This means tracking would not be consistent, and therefore the approach a bad practice. *File formats* are API dependent, but for every image recognition software it would be only logical to include high resolution pictures. Vuforia defines their files to be of type Portable Network Graphics (.png) or Joint Photographic Experts Group (.jpg) and to have from 8 bit up to 24 bit color depth. All files need to be smaller than 2 MB and .jpg files may not be in Cyan Magenta Yellow Key (CMYK) color model, meaning only the Red Green Blue (RGB) color model or grayscale images are allowed. As .png is a lossless, but well compressible file format, we chose to develop our image target with this file extension [Gro].

Vuforia employs a five star rating technique, which sorts pictures into five categories of machine readability. If every advice of the best practices has been followed, a full five star rating will be issued. This means, under perfect circumstances (see Section 2.4.1) there should be no way the marker doesn't get recognized. These stars help developers and designers to quickly understand their mistakes and adjust accordingly. For this thesis no lower that five star image targets were developed.

A so called feature is simply put an area inside the image, which has a lot of de-

tail for the search algorithm to find. Such features are for example corners of squares or triangles, but not circled or bent shapes. It is important to note that organic edges and soft corners are not considered features, as a rounded shape can not be identified properly. Thus it is best to use sharp and clear geometry for image targets. A simple way of understanding this, comes with the introduction of Quick Response (QR)-codes. These codes are a concatenation of usually black squares on a white background, as can be seen in Fig. 2.2. For our approach we used a random sample of the QR-Code generator provided by the open source software ARMaker by Lehner [Leh] in concatenation with custom self defined geometry for better understanding and human readability.

Figure 2.2: A randomly generated QR-code (left), where features are displayed as yellow crosses (right). It is easy to see, why vertices are best used for feature detection.

Features should be distributed evenly among the image target, as this ensures best tracking. Naturally, augmented models are steadier, if placed on top of a feature rich area. This task can be quite tedious, but defines how well tracking under motion will occur.

Repetitive patterns should be avoided, since it could easily confuse the search algorithm concerning 3D orientation. For example, a chess board has many features which could be tracked, evenly distributed and with good contrast. However each feature looks the same, meaning huge complications if we rotate the marker $90°$ multiple times. Compare this with the approach of unique QR-Codes, which when rotated, are still tracked perfectly.

Speaking about contrast, no feature could be detected, if the currently examined area is not contrast rich. As is widely known, areas with big differences in contrast usually mean a change of objects, textures or materials. Still, if there is only a small change, like e.g., a color gradient, the algorithm would not classify the area as containing different

objects, thus not identifying different features.

The Feature-Exclusion Buffer is a technique to better differentiate between various textures and is used once for each image target [Vufa]. It cuts off an 8% margin on each side of the square image to improve machine readability and to easily distinguish between different textures. As we still want this additional 8% to be used, we simply create a colored margin border the size of the Feature-Exclusion Buffer, which in order lets us use the additional space. This border may not have the same color as the background behind the marker, thus we decided black to be the best fit, as the finished marker should be printed on white paper.

2.4.3 Optimization and Evaluation

To evaluate an image target quickly it is possible to consider the grayscale histogram. As Vuforia converts every image to grayscale (to spare valuable processing time) before calculating its features, it is only logical to examine the the picture in gray. As can be seen in Fig. 2.3, it is better to have a color histogram that is wide and flat.

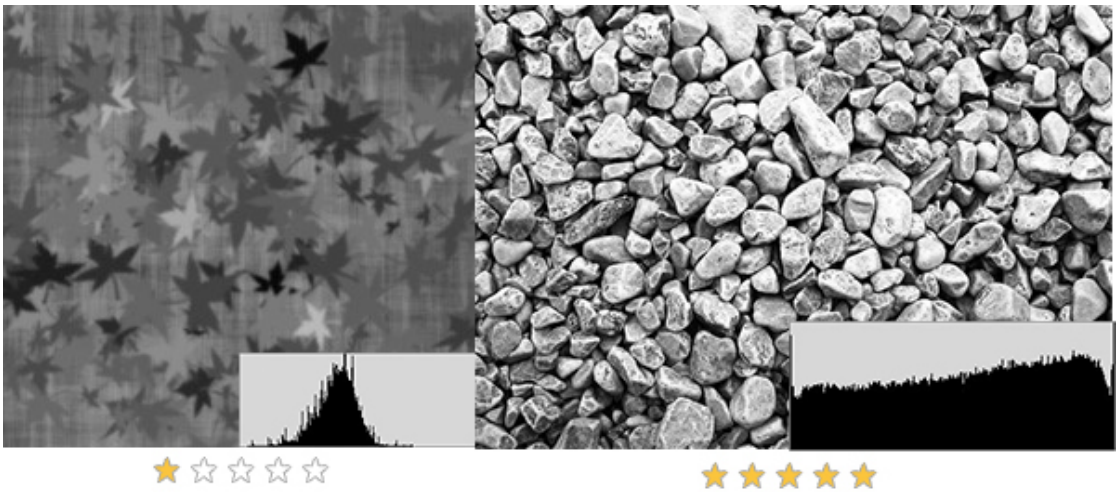


Figure 2.3: Comparison of contrast between two image targets. Bad (left) images are easily discerned from good (right) ones, due to a heterogeneous graysacle histogram. The star-ranking (bottom) depicts Vuforias' internal grading system judging the contrast. Image by Vuforia [Vufb].

Local contrast is almost as an important factor as global contrast. Simply put, if we zoom into a picture enough, we will usually see changes in contrast getting lower. To circumvent this, we use Local Contrast Enhancement [Vufb]. Many may already know this feature under another name: the Sharpen Tool, or even Histogram Equalization. It enhances, or sharpens, the examined area, which simply means increasing the color value. Pictures may not look as pleasing after this treatment, but it is an important tool for machine readability.

Of course, if markers are moved, the algorithm has to keep up with the tracking. This can be facilitated by using Device Tracking. This option enables the additional tracking of the cameras movement, meaning additional positional data, which in turn can be included for visual calculations.

As for our use case also the human body is a variable, and so for an optimal tracking the camera should be able to see all five markers (two on the shoulders, two on the hips, one approximately at the heart), be placed stationary or held quite stationary and the viewed body should not move to much. This ensures accurate tracking, similar to testing conditions.

CHAPTER 3

CARdiFlow: an AR Mobile App for Education

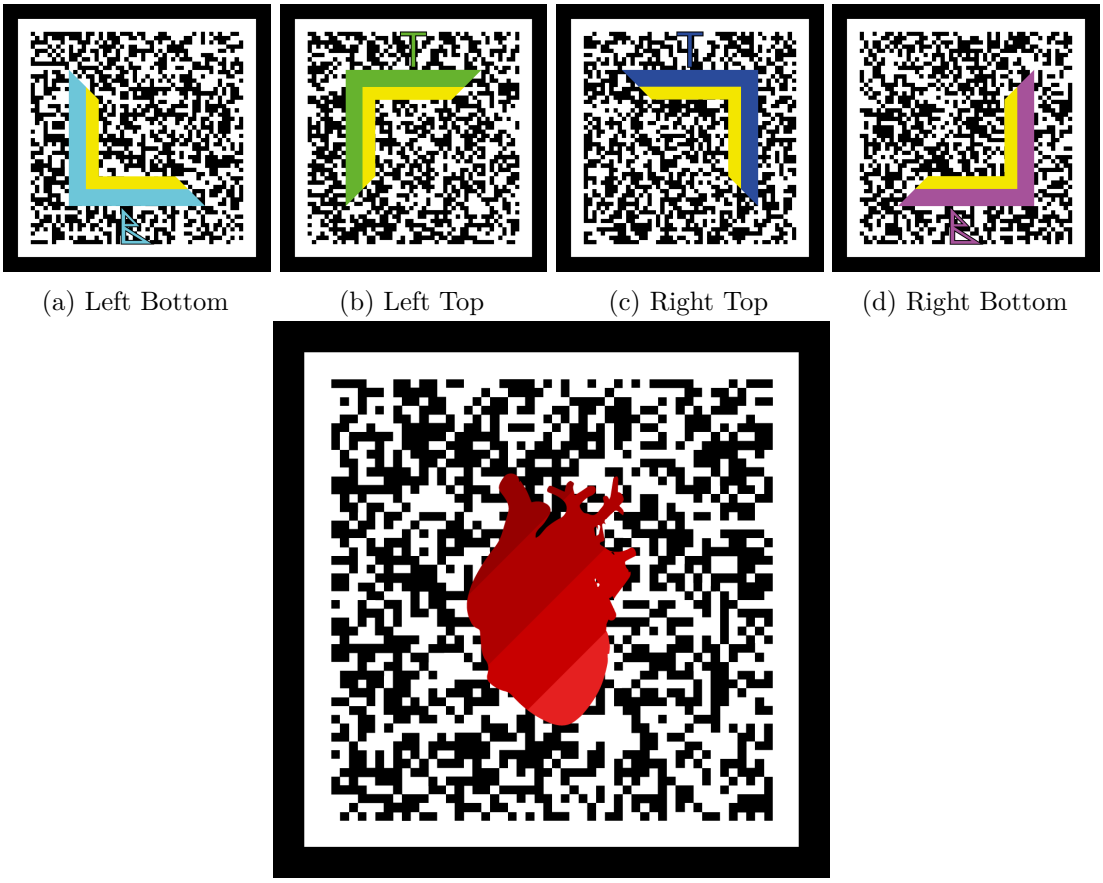
CARdiFlow is an Augmented Reality (AR) mobile application, displaying and teaching the cardiac blood flow for educational purposes. As the markers, needed for AR, went through multiple development cycles, so did the software — getting to the final tech stack of using:

- `Blender`, for the refining of the heart model, as well as simulation and generation of cardiac blood flow.
- `Unity`, for creating and visualizing the executable and bundling tasks into small achievable work packages.
- `Vuforia`, for implementing and delivering an authentic Augmented Reality experience.
- `C#`, as the programming language to connect the tech stack in an object oriented way.

An online repository with the implementation can be found by following this URL: `https://github.com/Zai-shen/CARdiFlow` [Jan].

3.1 Markers

Augmented Reality can be centered around many objects and patterns, the most stable approach being markers. Our markers consider all the physical properties from the recommendation list and incorporate all of the best practices we identified earlier. The Grayscale Evaluation was used for quick prototyping and local contrast enhancement was



(a) Left Bottom (b) Left Top (c) Right Top (d) Right Bottom

(e) Heart

Figure 3.1: Markers for *CARdiFlow* tracking. Each corner represents a quadrant on the upper body, the heart will display a high quality model of the human heart.

preferred for the final image target. *CARdiFlow* markers should be printed on rough, non-reflective paper, preferably equal or harder than 200 g/m^2. Lighting conditions should be at least dimly lit, motion of the human body should be reduced to a needed minimum. If these conditions are met, tracking for our markers should be reliable and fluid (Fig. 3.1).

We designed five markers, of which we thought, the outer four (Fig. 3.2) will be helpful for resizing the interactive model of the human heart. This was proven wrong, as there could not be found any scientific approach to resizing the heart in correlation to torso size. Also thorough research revealed, that the heart's size may depend on many different factors, but astonishingly closely matches any user's closed fist. Therefore, we implemented a slider functionality to resize based on one's custom size.

The center piece is responsible for displaying the heart, which will then be filled with content at runtime. The user is instructed to apply the marker via duct tape, two-sided

tape or other means onto their clothes, but could also easily hold it in their hand. Alternatively, it is also possible to place the marker on a flat surface and view it like an interactive 3D model, for a very stable and sterile learning opportunity.

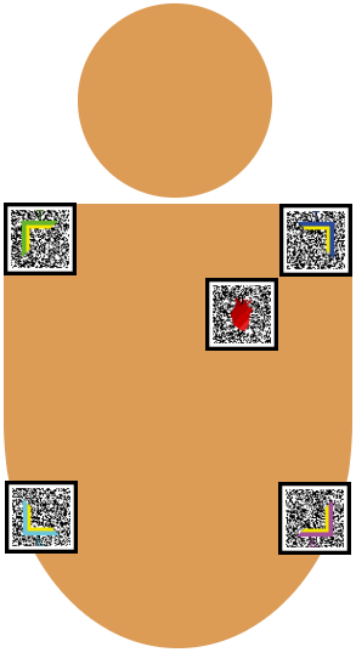


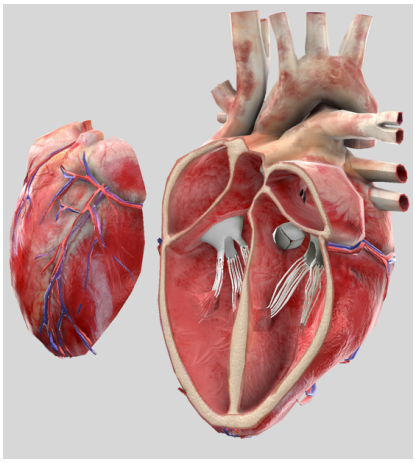
Figure 3.2: The four positional markers should be placed at the hips and shoulders, facing to the torso center. The heart marker should be placed approximately on the real world counterpart.

3.2 Simulation

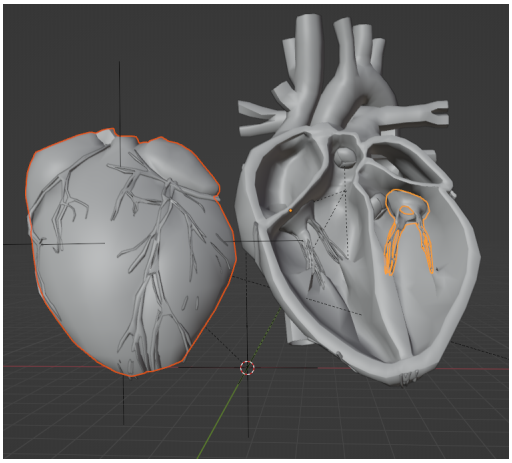
To simulate cardiac blood flow as close to reality as possible, we needed data structures fit for the task. In the upcoming sections we will talk about the structure of an ideal 3D heart model, conditions for realistic blood flows, helper functions for said blood flows as well as data formats for exporting generated flows.

3.2.1 3D Model

It is of utter importance that the different meshes, which will be displayed on screen, are as simple as can be. Therefore, we decided on a low polygon model, which still conveys a realistic feel through textures and materials. We came up with a model [CGZ] (see Fig. 3.3a) from the popular modelling website `Turbosquid.com` [Tur] which fulfills these requirements pretty well, but nevertheless there were some changes needed. As the model was given without animation in one single state, as can be seen in Fig. 3.3b, we



(a) Even though the topology of the mesh is low on polygons, the high quality textures and materials hide that fact quite nicely.



(b) The model consists of different objects, but in a single state. The selected (orange) object is only available in its open form.

Figure 3.3: The heart model, textured (a) and untextured (b).

needed to close various valves visually as well as programmatic, so that blood would not flow out of the container it was designed to be in.

3.2.2 Blood Flow

Simulation of blood flows, or liquids in general, can be achieved in various different ways, be it runtime or pre-baked. We settled for `Blenders`' built-in fluid dynamics, which allows us to pre-bake the animation into separate files, which are then converted and loaded in `Unity` at runtime, resulting in a more performant display of blood flows, while still remaining true to detail.

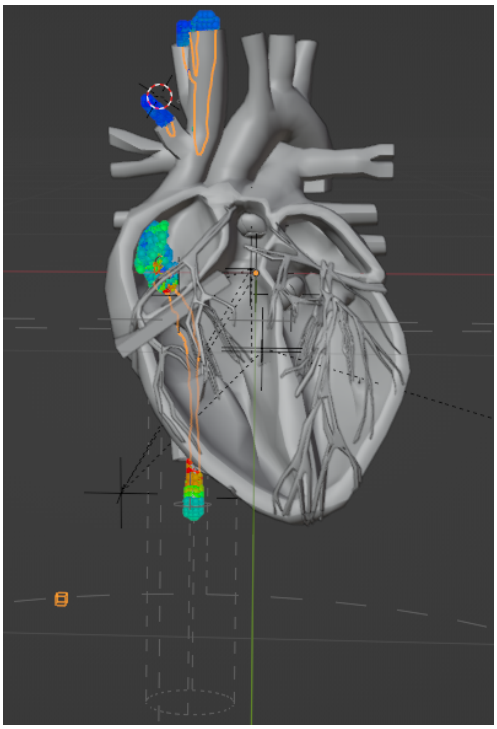
Blender Fluid Physics

Fluids in `Blender` are generated by either some of the external paid plugins, or their built-in fluid physics system. The native built-in approach seemed fit for the task and will also see long time support, so we decided to rely on it.

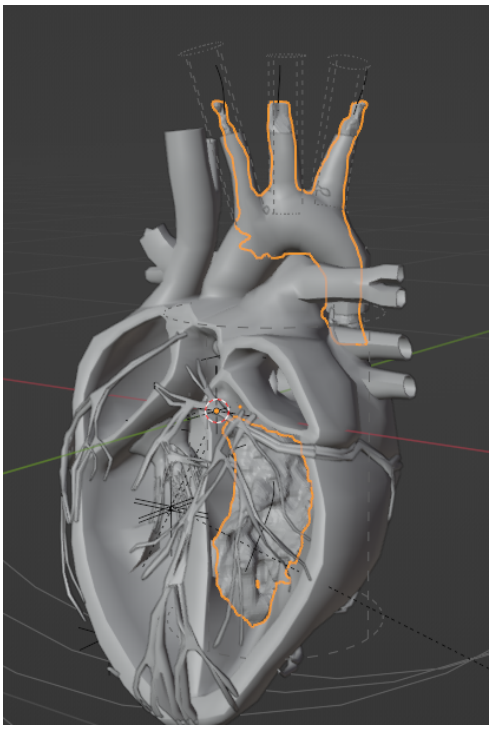
For a fluid system to work in `Blender` one needs to implement at least two distinct modules:

- A domain, which defines the bounds and resolution of the simulation volume.
- A flow, which will define and emit the fluid.

Furthermore several effectors are required to make the fluid interact (e.g., collide) with objects in the scene. Additionally we used different kinds of force fields (see Fig. 3.4b),



(a) Blood flowing into the right atrium, represented by different particles and their velocity, heatmapped.



(b) Blood flowing out of the aorta, visualized as non textured closed meshes. Note the gray lines depicting force fields.

Figure 3.4: Blood flows as represented in `Blender`, with particles displayed (a) or fully meshed out (b).

which allowed us to really cherry pick different flow configurations by simply manipulating flow directions through simulated force.

We decided on a function based hierarchy (see Fig. 3.5) to be able to quickly manage and iterate on the different flows. Ideally only internal effectors would be needed, meaning we would only need to mark the hearts walls as colliders, but sadly this was not sufficient, as even after multiple approaches the blood would still find a way to drip or clip through the heart. Therefore, we had to manually mock thicker walls by creating invisible effectors covering the outside of the model.

Realistic Blood

To achieve realistic feeling blood we had to modify the fluid domain in `Blender`. As can be seen in Fig. A.1 we settled for a resolution of `100` units, meaning the cube enclosing the simulation would be subdivided into `100` equal sized chunks, each calculating a cell of fluid. We found this number to be the best compromise between calculation speed and visual complexity, but with better equipment or more time this number could easily be

Figure 3.5: A function based hierarchy facilitates quick scene management. SimBounds represents the simulation domain.

stepped up a notch. The time scale indicates how fast the simulation will tick, and was chosen per blood flow for an appealing transition. The amount of time steps determines how often the simulation will be calculated per frame of the animation. Higher numbers result in more accurate simulation and better handling of narrow geometry, but also in higher computational effort and compile times.

`Blender` supports two different fluid simulation methods: Fluid Implicit Particle (FLIP) and Affine Particle-in-Cell (APIC), which are just two different algorithmic solvers. While FLIP produces more fine grained and detailed, but also chaotic simulations, APIC delivers smooth and slower, but also more reliable simulations. In Fig. A.2 we can observe, that it is even possible, while still choosing FLIP, to set an interpolation number — the FLIP ratio. This value sets how much velocity is used when updating liquid particle velocities, with a value of `1.0` resulting in completely FLIP calculations. A value of `0.87` was determined to be a good mix between turbulence and visual clarity of the animation.

When it comes to fluids, we need to account for their viscosity. As seen in Fig. A.3 this can be done in `Blender` via the attributes viscosity and diffusion. While higher values result in more viscous fluids, a value of zero still applies some viscosity. Sadly `Blenders` viscosity amounts are not coupled to any scientific method of measuring viscosity like kinetic ($\sim$ `2.74` $mm^2/$`s`) or dynamic ($\sim$ `2.87 mPa-s`) at `35°C` [Gmb]. To set these values into correlation see Table 3.1. While diffusion makes use of measurable attributes for viscous fluids, in our tests it did not seem to affect the fluid at all.

While enabling us to a fully iterable and reproducible approach to fluid simulation, we found `Blenders`' built-in simulation rather tedious and in many cases error prone. We had to manually direct and drag the blood flows through their path via force fields, namely wind and curve guides. This had to be done, as liquid would otherwise clip through little crevices or even solid material, even after the usage of transparent effectors. In future work we would recommend outsourcing this kind of work to external artists, or using different, more precise solvers.

Fluid	**Dynamic Viscosity** $[cP]$	**Kinematic Viscosity** cST	**Temperature** $°C$
Air	0.018	13.9	27
Water	0.894	0.894	25
Water	1	1	20
Ethanol	1.074	1.36	25
Mercury	1.526	0.11	25
Honey	5000	3500	25

Table 3.1: Viscosity of different well-known substances [Incb]. (1 milliPascal-second [mPa-s] = 1 centiPoise [cP] = 1 centiStoke $[mm^2/s]$).

3.2.3 Animation Export

Finding an up-to-date approach for exporting files was no easy task, as different file formats all had their (dis-)advantages.

The first an simplest method would be to use `Blender`'s intern file format `.blend`, as `Unity` supports some of these external 3D model software formats. This would leave us with the ability to quickly change the source file and retain any changes, without having to manually update anything in the `Unity` editor. Sadly this approach was deemed unusable, as absolutely everything in the `.blend` scene would get imported, meaning we would have to delete objects again. In addition, animations were not baked — leaving us only with the starting geometry of several droplets without any animation.

The next approach identified the Graphics Language Transmission Format `(.gltf)` and its update `GLTF2.0` as a viable candidate. Not only does it support animations over time, but it also enables to still identify separate objects after a strong compression. It is even possible to ship it as a single `.glb` file, a binary form, which includes object textures instead of referencing them as external images. Also this format was not able to be chosen, as it can only support animations, which are based on a skeletonized model. Since we are simulating fluids, it is logical to assume this to not be feasible, as each and every drop of blood would need to be its own (skeletonized) model and corresponding file.

A long time ago it was the state of the art to export a single Wavefront Technologies OBJ geometry format `(.obj)` file with a single Material Template Library `(.mtl)` file per frame of the animation, so one would only have to iterate over all frames for a full animation. This approach is similar to how animated films were created at the time, only that each frame was drawn per hand (see Fig. 3.6) and not a complete 3D file. This seemed to work at first, but was deemed hugely inperformant after certain test runs. Since we went for at least `30` frames per second (FPS), and at least `2` seconds of animation per flow, we would need in sum `60` `.obj` and `.mtl` files to be loaded for every `2` seconds.

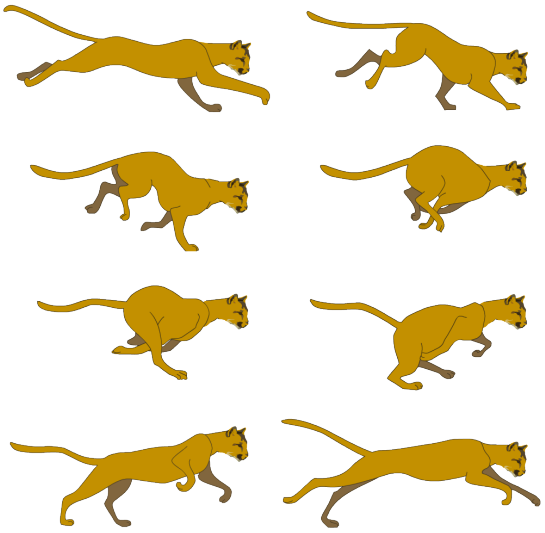


Figure 3.6: Classic sprite sheet animation used for the 8 frames of a lion running.

The problem with this approach is, that it is raw, without optimizations, or compression. We cannot make use of smaller files, like with `.gltf` compression (see Fig. 3.7) mentioned by [Lee19], therefore, we would need to load `1.1 GB` of data from the disk for each animation. This is nowhere near being possible as of today's standards, especially for mobile devices. And even if the data could be pre-loaded, it would still be `6` animations, therefore, around `6.6 GB` of Random Access Memory (RAM) which would have to be allocated only for this purpose.

Finally, we tested the Alembic `(.abc)` interchangeable computer graphics file format, which was announced at SIGGRAPH 2011 and promises great performance. Similarly to `GLTF` it supports the export of whole scenes, but is not bound to skeletonized models. This enables us to share optimized and compressed meshes, or even whole animations, quick and clean. With this approach we managed to reduce the file size of an animation to the average of ≈`45 MB`, for a total of roughly `268 MB`, even when retaining stable `30 FPS`. This provided an improvement of `1 GB` of file size in comparison with the `.obj` approach, but also having a single file and no external image data proved to be very useful. `Unity` supports the import of `.abc` files via a library, available for free at their asset store.

3.3 Visualization

With the simulation done, we needed the flow data to be visualized. For this, there are again multiple different frameworks, which each offer their advantages and disadvantages.

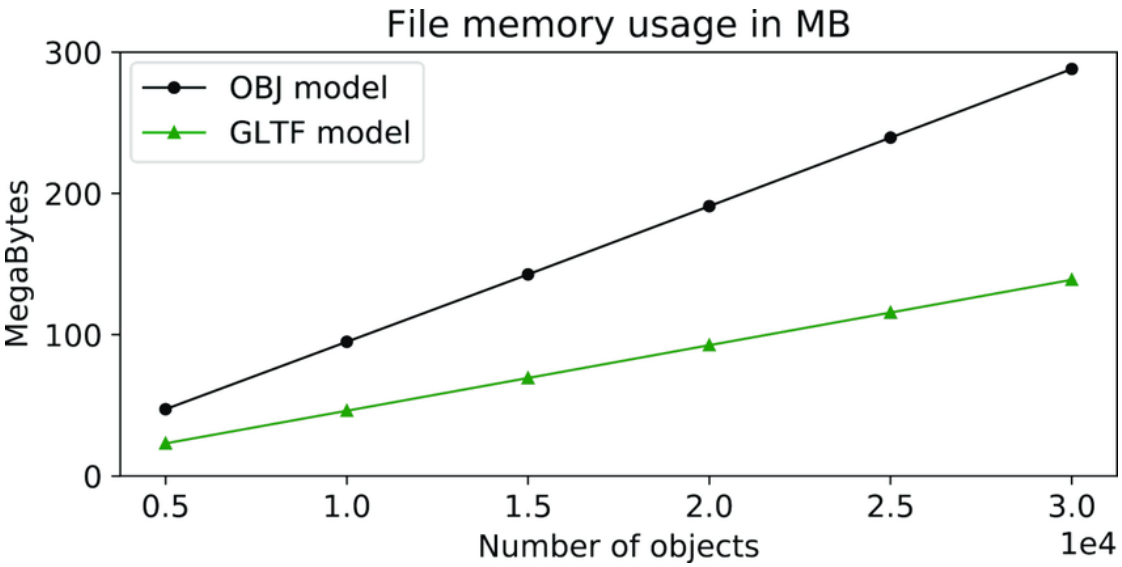


Figure 3.7: The compression of `.gltf` is far superior to raw `.obj`. Image by Fia Sutton [Sut].

3.3.1 Unity

We settled for the `Unity` engine, as it provides easy manipulation of objects, a load of predefined components useful for rendering, broad compatibility to various platforms, including android, as well as a stable and future proof editor with their promise of long time support.

`Unity` offers four different kinds of render pipelines, which are in different states of development, as well as platform support [Tecb].

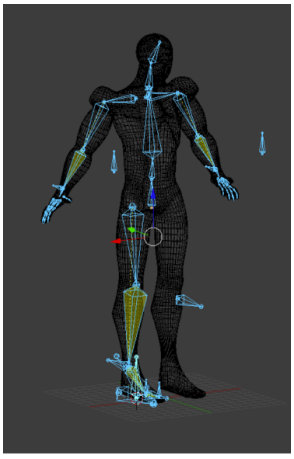
- The Built-In Render Pipeline is `Unity`'s standard way of visualizing objects and offers the most stable performance, the best platform support as well as the biggest support for asset packages loaded from `Unity`'s asset store.
- The Universal Render Pipeline (URP) is believed to be the next contender to `Unity`'s standard pipeline, but still lacks in stability. Though the light calculations of the URP are a lot faster and more accurate, they are limited to 32 lights per camera, on mobile devices. This proved to be of no concern for this project, but could certainly be of importance for further work.
- The High Definition Render Pipeline (HDRP) offers the most sophisticated approach to rendering in `Unity` by also supporting ray-tracing, although it has the highest performance requirements and smallest platform support. It was developed for AAA game development, real time graphics and the movie industry and therefore is not yet supporting mobile devices.

- A Custom Render Pipeline can be created, as Unity supports this with an interface to the Scriptable Render Pipeline (SRP) API, which is the same one powering URP and HDRP. This approach is most fit for big corporations with a dedicated team developing only this pipeline, as it offers the best customizability, but also needs a lot of work to be put into it.

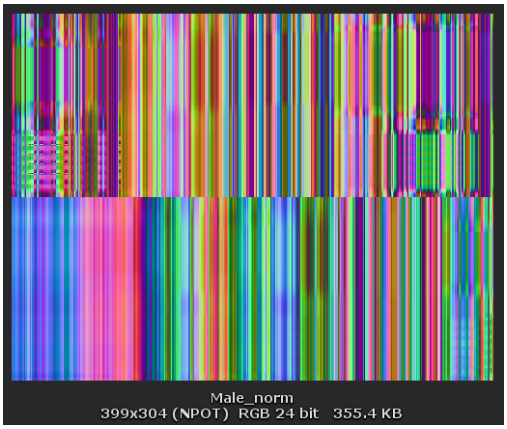
We found the Built-In Render Pipeline to be the best fit for this project, as it offers the greatest stability while also offering the best platform support.

3.3.2 Vertex Animation Textures

After the project was set up in Unity and the Simulation developed in Blender, we just needed to import it in a working format. As previously discussed, the Alembic file format (.abc) was used for export and import of the simulation. Unity does not support this out of the box, but with the help of the Alembic for Unity package, which is available for free on the asset store, we managed to get the import done. Although the visualization of the Alembic animation worked just fine when built for PC or in Unity's Editor Mode, it would not work on mobile. It seems fluid simulations on mobile devices in general are hugely under-utilized, and therefore there is nearly no support available for the task. The only alternative seemed to be to simulated the fluids dynamically via an asset at realtime. Obviously this approach would not even nearly deliver the level of detail we would need for cardiac blood flow, and therefore we found ourselves searching for a novel approach to visualization of fluids on mobile devices.



(a) A typical visualization of a rigged human body. Image by Open3DModel.com [N.Aa].

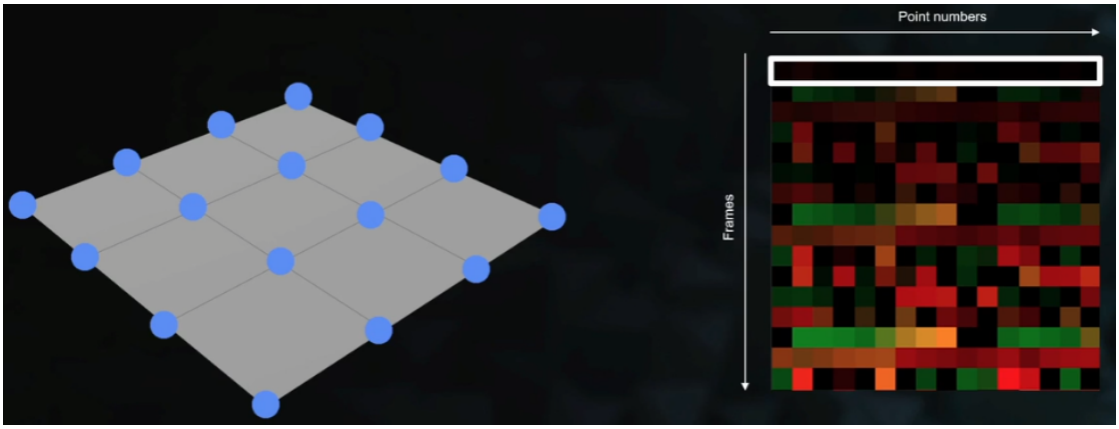


(b) A sample of a Vertex Animation Texture. Image by miro.medium.com [N.Ab].

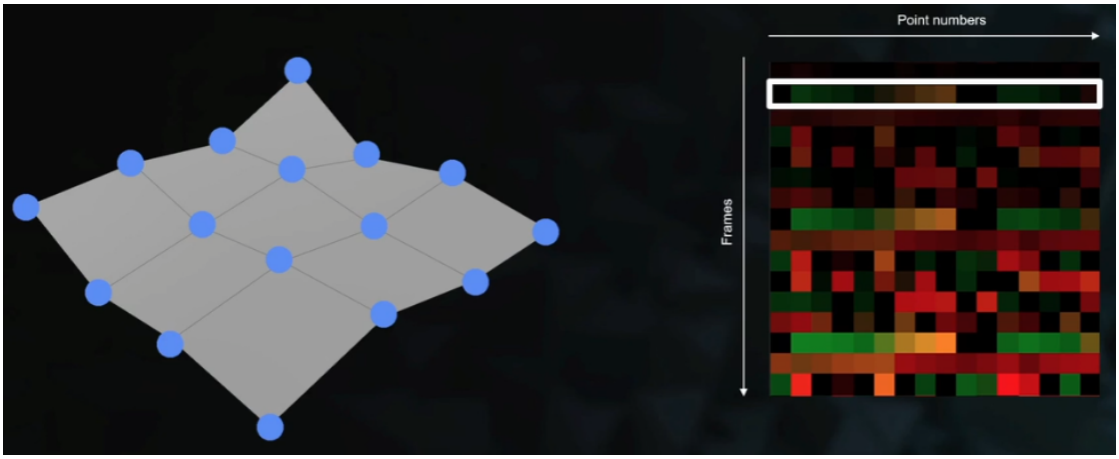
Figure 3.8: Each frame step the mesh gets transformed, as can be seen before (a) and after (b).

The solution to this dilemma was found in Vertex Animation Texturing (VAT) (see Fig. 3.8b). Usually, when animating in a game engine like Unity, we make use of

the bone workflow (see Fig. 3.8a). This means we have a fully rigged model with a skeleton-like bone infrastructure, which is then easily modifiable by translating or rotating said bones, to transform the bound mesh and its vertices. This will not work, if the mesh should be broken, or new geometry formed, like it is the case with fluids, e.g., a simple drop of blood can split into multiple, a flow can be teared apart. Vertex animation textures save the animation, meaning they save the mesh and its transformation for each frame, as can be seen in Fig. 3.9, which is then exposed in a time variable to let us control the state. This of course means the textures will of huge file size, if we don't set certain limits. That is why we decided to stay at `30 FPS`, each animation taking a maximum of `2` seconds to complete, ensuring a performant and well visible display of our blood flow. Additionally, these textures are loaded and displayed by the Graphics Processing Unit (GPU) instead of the Central Processing Unit (CPU), which increases performance immensely.



(a) The mesh before transformation (left) and the frame step in the texture (right).



(b) The mesh after transformation (left) and the frame step in the texture (right).

Figure 3.9: Each frame step the mesh gets transformed, as can be seen before (a) and after (b). Images by Simon Verstraete [Ver].

Unity does support these VA textures and is even able to display them on mobile devices. Therefore, we found a novel approach to displaying complex and fast moving fluids reliably on mobile devices, which greatly improves visual quality in comparison to traditional approaches.

Now we only need a way to convert our Alembic files to VAT, which is done by Gaxil's [htt] great and free asset called Unity-AlembicToVAT. With this tool it is easy to exchange our files, and it even provides shaders which are able to iterate over the VAT time variable.

3.3.3 Vuforia

Vuforia is a quick and efficient framework, which enables and supports Extended Reality (XR) development around the world. Through its use we were able to implement Augmented Reality functionality with many fast iteration cycles into our Unity application.

To be able to modify Vuforia's general settings, one has to locate the file called VuforiaConfiguration in the Resources folder, which is of tremendous importance. Firstly we had to get hold of a developer account, which in turn comes with a Application License Key, which activates all AR functionalities of the API. Secondly we set the maximum simultaneously tracked image targets to 1 instance only, to spare valuable computational time and to not let the ARCamera recognize more targets per accident. The configuration also offers a boolean field for managing, if Vuforia should track the device pose, which theoretically should improve the precision and performance of tracking by accounting for Degrees of Freedom (DoF). These enable the device to catch multiple angles of rotation and position, and should therefore add to the tracking precision, but in our case only eliminated previously supported devices, which were only capable of 3 DoF instead of 6 DoF.

The Main scene was supported by Vuforia, by creating a pre-defined GameObject named ARCamera, which can be seen in Fig. A.4. It is composed by creating a standard camera game object in Unity and adding a component each for the Vuforia Behaviour and the Default Initialization Error Handler. It is important to note we set the cameras' near clipping plane to a very small value, enabling the user to be really close to the heart model and still being rendered a correct image of the focus area without it being culled pre-maturely. Following this we created 6 objects (see Fig. A.5), each containing a different cardiac blood flow, displaying and animating the Vertex Array Texture. To keep these GameObjects in place, we parented them all to the same GameObject, the ImageTarget_Heart, which would let us simply translate and rotate the parent, without having to repeat each step 6 times. Furthermore the Vuforia Image Target is responsible for displaying its children whenever the given marker (see Fig. 3.1e) is being detected by the ARCamera. This did lay the base for a complete AR experience.

Vuforia, as of Version 9.8.8, though has its caveats. Firstly, it was developed as a

standalone product and only after seeing high demand it was ported as an implementable API to `Unity`. This means, that it does not support `Unity`'s scene management out of the box, which was a big encumbrance, because the augmented reality camera, with its feature recognition, is activated at all times, resulting in huge battery drain and an always active background override. To circumvent this we based our development of a script (see Lst. B.1) on Caiovm's findings [cai]. This works just as intended and lets us access the camera management.

Furthermore, `Vuforia` would not let the application start with the `Menu` scene, as it instantly switched to the scene with an active AR camera. This odd behavior was prevented by the very same script from above, but also by adding a `Vuforia Behaviour` component to the main camera and disabling it. It seems `Vuforia`'s initialization process searches for this component, and by letting it be disabled, it gets ignored, therefore, allowing for normal scene flow. To note is, that the `VuforiaConfiguration` offers a variable called `DelayedInitalization` which should handle this behavior, but does not actually work.

3.3.4 Application features

Main Menu and Options

When it comes to developing a mobile application, the first screen an user usually sees is the main menu. We decided to go with a red theme (see Fig. 3.10), conveying a medical station feeling, supported by a responsive design, which can easily be ported to different screen resolutions.

Figure 3.10: A responsive design and red background support user engagement.

For this we made use of `Unity`'s new User Interface Toolkit, which enables us to define UI elements via `.uxml`, their internal take on `.xml`, and style these objects via `.uss`, corresponding to regular `.css`. The menu includes a button leading to a tutorial, which provides users with further information (if not already known by following the evaluation file handed out in `.pdf`) and an options panel, which lets users define various graphical settings, therefore also enabling support of low-end mobile devices.

Interactive Toggle Buttons

To let users interact with the runtime application itself, we decided to provide an intuitive user interface. After thorough research we determined the lower left side of any smartphone's screen to be best fit for a panel display of four toggle buttons (see Fig. 3.11), as users will not unintentionally touch them while handling the complex dynamics of an augmented reality application. Each one of these toggle buttons enables a different visual functionality of *CARdiFlow*, while often also providing further panels to precisely foster a users gained knowledge.

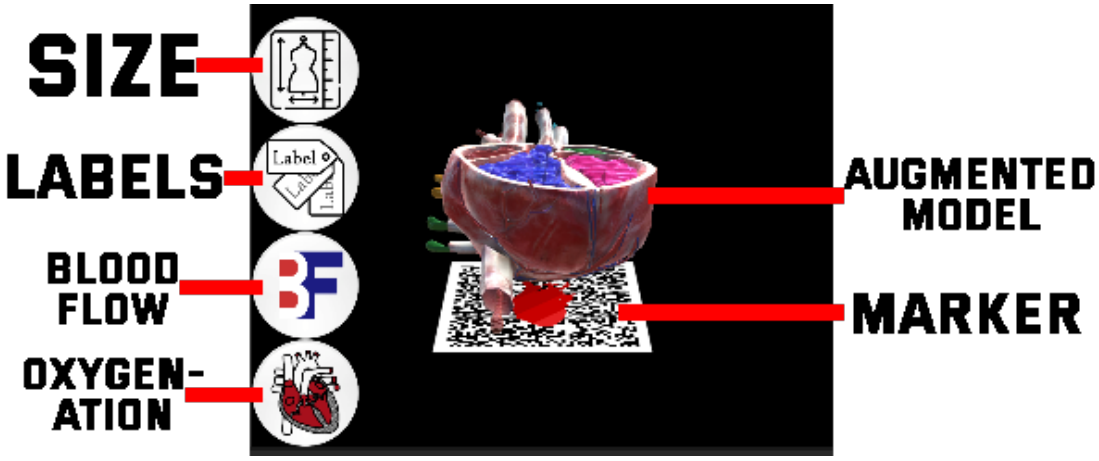


Figure 3.11: *CARdiFlow*'s user interface provides four toggles, enabling to change heart size, display of labels, display of blood flows and the state of blood oxygenation.

Heart Resizing

The generated visualization of the users heart is only immersive and therefore viable, if it comes close to reality. Therefore, we provide the ability to resize the heart with the slider seen in Fig. 3.12.

The iteration steps are provided by centimeters, with a default size of `12 cm`. The user is instructed, either already read in-app or in the study instructions, to set this parameter roughly to the height of ones hand size, which resembles the hearts true size. This can be achieved by measuring ones hand, either with a ruler, or by placing the hand in front of the camera, thus roughly approximating the size. In code (see Lst. B.3) this is done by

Figure 3.12: By enabling the heart resizing toggle a slider is displayed, letting the user adjust the hearts size in correlation to its height in centimeters.

awaiting `Unity`'s inbuilt dynamic slider callback to resize the objects current size (the objects' transform local scale) to the requested value.

Label Display

To facilitate a better learning experience, the user is able to display labels all around the heart. This is done by enabling the label display toggle, and then selecting the desired granularity of labels (see Fig. 3.13). This was implemented by creating the labels before compilation, tagging them with a descriptive tag for its category (*SPARSE, MODERATE, PRECISE*) and deactivating the correlating game objects beforehand. When the user changes the display at runtime it invokes a call to the `LabelController` (see Lst. B.2) to search for the given objects by tag and dis-/enable them.

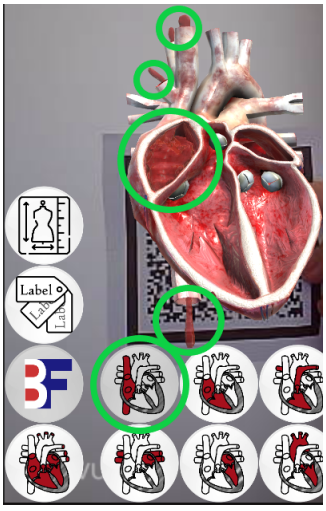
Figure 3.13: By enabling the label display toggle four different toggles are displayed, letting the user adjust the granularity of labeling they desire.

Flow Selection

Being able to display different flows is of upmost importance, therefore, we implemented a toggle, which enables eight further toggles (see Fig. 3.14). Six of them will enable any one separate flow area, be it left/right atrium, left/right ventricle, aorta or pulmonary arteries. The other two will disable all flows, or enable the dual flow mode, which activates both (left and right atrium) ⇒ (left and right ventricle) ⇒ (aorta / pulmonary valve) simultaneously.

To achieve this we implemented a system for managing blood flows, controlling each and every flow, their duration and their animation. The underlying scheme for controllers in `Unity` is usually done by the Singleton pattern, therefore, we adapted this workflow.

(a) By enabling the flow areas toggle eight different toggles are displayed, letting the user select the blood flow they want to render.



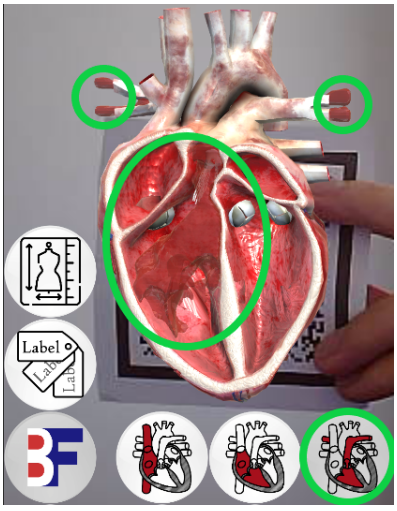


Figure 3.14: The flow area toggles (a) control the displayed and animated flows. Here we see the animation of the right atrium (b) and the pulmonary valve (c).

Singletons are a widely spread but also controversial approach for handling a single instance of an important object, but in `Unity` they are an essential way of handling static access. While a Singleton more tightly packs the code base, it allows for quick and global access — this is exactly what is needed for handling our blood flows. It is also in direct contact with the `ValveController`, responsible for enabling and disabling the valves connecting different flow areas, therefore handling the tricuspid-, pulmonary-, mitral- and aortic valve. The `FlowController` is also based on the Update pattern, calculating and managing flows every frame, if the heart marker is being tracked, as can be seen in Lst. 3.1.

Listing 3.1: Part of FlowController.cs

```
1     // Update is called once per frame
2     void Update()
3     {
4         if (!isTracked)
5         {
6             return;
7         }
8         durationTimer += Time.deltaTime;
9         if (durationTimer >= duration)
10        {
11            durationTimer = 0f;
12            if (ActiveFlowsExist())
13            {
14                HandleFlowing();
15            }
16        }
17    }
```

Oxygenation Display

A toggle is provided to understand in detail, which areas are carrying oxygenated blood, and which are carrying deoxygenated blood. This is toggled by de-/activating the toggle seen in Fig. 3.15. In code this is done by accessing the `FlowController`'s flows by previously set indices and changing their materials color to the desired value (see Lst. B.4).

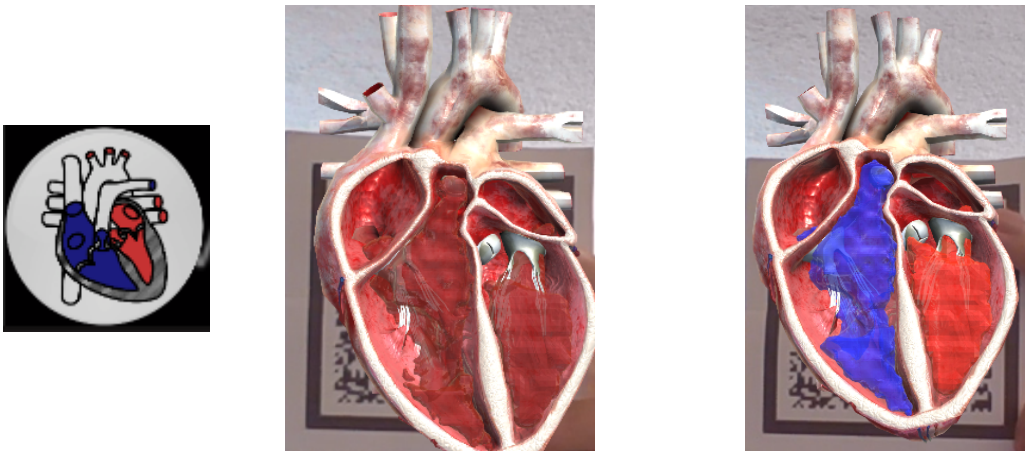


Figure 3.15: By enabling the oxygenation toggle (left) the different states of blood oxygenation are displayed by changing the flow colors (right) to red, indicating oxygenated blood, and blue, indicating deoxygenated blood.

CHAPTER 4

Results

4.1 Evaluation Method

To confirm that our approach provides benefits in the learning process of students as compared to traditional approaches (i.e., static illustrations with additional labeling), we need to evaluate our developed application. Usually this is done by providing different source materials to the test subjects, evaluating the group means and comparing them. This implies that an approach should be used, which can easily facilitate the comparison between the application's users, i.e., the experimental group, and the users which got the traditional learning material, i.e., the control group. We found A | B-Testing (also known as Split Testing) to be most suitable, as it is already commonly found in User Experience (UX) cycles and also starts being seen more in Agile Software Development cycles [ØL15]. The decision, to only incorporate static and AR content was based on the widespread use of static illustrations for anatomical education, e.g., in school, but also on practical reasons, such as the participant count. To assess against, e.g., on screen visualizations, would require additional participants in a full-fledged evaluation, which would not be feasible for the course of this thesis. Future work, with more participants and budget, could also include testing against animations, or even voiced tutorial videos. To implement this we provided two different learning structures, both hosted on an online storage for quick and seamless distribution. We designed separate instruction `.pdf` files and allowed the application to be directly downloaded via network, which also further increased prototyping and troubleshooting loop times, as different versions of Android each come with their own caveats.

4.1.1 Brief Examination

We provided the users with thorough introduction, a page for demographic information followed by confirmation of consent and the tutorial. Users then found the first work part of the evaluation: a short examination. We implemented a mix of single choice questions,

ordering- and marking tasks, to determine the students amount of knowledge earned during the learning phase. The questions were explicitly formulated to not give away correct answers and were based on the hearts blood flow as well as anatomy.

4.1.2 Adapted System Usability Scale

Following the examination, we designed a page to account for user experience. We aimed to answer:

- *How much did they like the experience?*
- *What was contributing to a good understanding?*
- *What exactly, if any, was a hindrance?*

To achieve this we mixed our questions with the well established `10` items of the System Usability Scale (SUS) developed by John Brooke in 1995 [Bro95]. This Likert scale allows one to evaluate effectiveness, efficiency and satisfaction for a wide variety of products and services, including hardware and software, which is exactly what we needed. The traditional SUS consists of 10 likert scales of alternating implications:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

We expanded the standard questionnaire (see Fig. 4.1) by three entries, specific to the developed application and targeting the learning effect:

11. I found the visualization to give me enough amounts of information about blood flows.

12. I found the visualization to give me enough amounts of information about heart structure.

13. I thought it was enjoyable to use this system.

The latter were added as we found they would best fit the scientific questions answered by this thesis. They enable us to answer targeted questions which are left unanswered by the traditional SUS for more detailed information about blood flow and heart structure. With this approach we got a comparison between both the traditional learning systems, i.e., static visualization, and modern learning systems, i.e., our AR application, based on one number, the SUS (`0-100`) score. Including the examination, it would already be possible to conduct results, but we wanted to account for every possible aspect.

4.1.3 Valuable User-Specific Input

Last but not least we wanted to also include user specific information. Therefore, we implemented open questions, asking them:

- How much time did they spend with the learning system?
- What, if anything, would they change in the given visualization?
- What other thoughts did they have, which could be useful for the project?

With this valuable information it should be easy to address and change features, which could then be developed in future work.

4.2 Evaluation Results

4.2.1 Raw Data

We conducted this study with a sample size of `n = 26` people, each from different socio-ecological and socio-economical standing points. `12` of them identify as women, `14` as men. There age distribution splits up in:

- `20-25:` `9` participants
- `25-30:` `10` participants
- `30-35:` `5` participants
- `35+:` `2` participants

We found 2 users to be doctors of human medicine and therefore knowledgeable in the medical domain, while 24 users determined to be laymen. The study was conducted online, due to the world-wide pandemic that is COVID-19, which in turn enabled the subjects to take their time and fully concentrate on the tasks given.

The exam we discussed in Section 4.1.1 was testing the students memory after being exposed to either traditional or modern learning method. The five tasks (here denoted as Q1 to Q5) were testing for how well the visualization aided in this process. A quick summary of the tasks at hand:

Q1. [*Size*] How big should the heart roughly be compared to your body? (single choice)

Q2. [*Order*] Order the blood flows, according to their inflow order. (max. 6 points)

Q3. [*Naming*] Name the flow areas by matching words to pre-defined lines. (max. 6 points)

Q4. [*Anatomy*] Mark the aortic valve.

Q5. [*Anatomy*] Mark the inferior vena cava.

In Table 4.1 we observe the exam results for the experimental group (Tab. 4.1b) and the control group (Tab. 4.1a). Additionally, we asked participants to time their usage of the learning materials in hope of connecting longer interaction times to better fostering of acquired information. This can be seen in Table 4.2.

ID	Q1	Q2	Q3	Q4	Q5
⋆ 1	yes	6	6	yes	yes
2	yes	6	6	yes	yes
3	yes	0	6	no	no
4	no	3	6	yes	yes
5	yes	6	6	yes	yes
6	yes	2	6	no	no
7	yes	6	6	yes	yes
8	yes	6	6	yes	yes
9	yes	6	6	yes	yes
10	yes	6	6	no	yes
11	yes	0	2	no	yes
12	yes	6	6	yes	yes
13	yes	6	6	yes	yes
$\sum$	12	59	74	9	11
$\overline{x}$	0,92	4,54	5,69	0,69	0,85
σ	0,27	2,31	1,07	0,46	0,36

(a) Experimentation group (EG) results.

ID	Q1	Q2	Q3	Q4	Q5
⋆ 14	yes	6	6	no	yes
15	yes	6	2	yes	no
16	yes	6	6	no	no
17	yes	6	6	no	yes
18	yes	2	0	no	no
19	yes	4	2	no	no
20	yes	6	6	no	yes
21	yes	0	6	no	no
22	yes	6	6	yes	no
23	yes	0	6	no	no
24	yes	6	4	no	no
25	yes	6	6	no	no
26	yes	6	6	yes	yes
$\sum$	13	60	62	4	4
$\overline{x}$	1,00	4,62	4,77	0,31	0,31
σ	0,00	2,27	2,01	0,46	0,46

(b) Control group (CG) results.

Table 4.1: Exam Results of the different groups. Note ID 1 & 14 (also indicated by a star and highlighted in red) are medical professionals. The EG worked with *CARdiFlow*, the CG with traditional learning means.

The raw data for the System Usability Scale tests can be seen in Table 4.3, while the aggregated SUS results can be found in Figures (Fig. 4.1 & Fig. 4.2) as Likert scales. It has been recorded as discussed in Section 4.1.2 and shows even on a quick glance that the experimentation group has had a very positive change in experience. Especially by focusing on the medical professionals in either group, we are able to see the sheer size of positive impact the modern approach (perfect score of 100) has had on them, compared to the traditional (score of 72.5). Note that missing values in the Likert scale were interpolated to the default value, being `3`, as per definition of the SUS [Bro95].

ID	Interaction Time
⋆ 1	20 minutes
2	15 minutes
3	5 minutes
4	15 minutes
5	15 minutes
6	10 minutes
7	20 minutes
8	10 minutes
9	15 minutes
10	30 minutes
11	10 minutes
12	15 minutes
13	20 minutes
$\overline{x}$	15,38 minutes
σ	6,03 minutes

(a) Experimentation group (EG) interaction time.

ID	Interaction Time
⋆ 14	5 minutes
15	10 minutes
16	10 minutes
17	20 minutes
18	10 minutes
19	5 minutes
20	15 minutes
21	10 minutes
22	5 minutes
23	10 minutes
24	15 minutes
25	5 minutes
26	10 minutes
$\overline{x}$	10,00 minutes
σ	4,39 minutes

(b) Control group (CG) interaction time.

Table 4.2: Interaction Times of the different groups. Note ID 1 & 14 (highlighted in red) are medical professionals. The EG worked with *CARdiFlow*, the CG with traditional learning means.

ID	SUS
⋆ 1	100
2	82,5
3	85
4	50
5	67,5
6	90
7	85
8	65
9	82,5
10	75
11	82,5
12	92,5
13	87,5
$\overline{x}$	80,38
σ	12,67

(a) Experimentation group (EG) System Usability Scale (SUS) value.

ID	SUS
⋆ 14	72,5
15	67,5
16	45
17	30
18	15
19	72,5
20	55
21	82,5
22	82,5
23	85
24	47,5
25	55
26	75
$\overline{x}$	60,38
σ	20,73

(b) Control group (CG) System Usability Scale (SUS) value.

Table 4.3: System Usability Scale (SUS) values of the different groups. Note ID 1 & 14 (highlighted in red) are medical professionals. Values higher than 68 are above average.

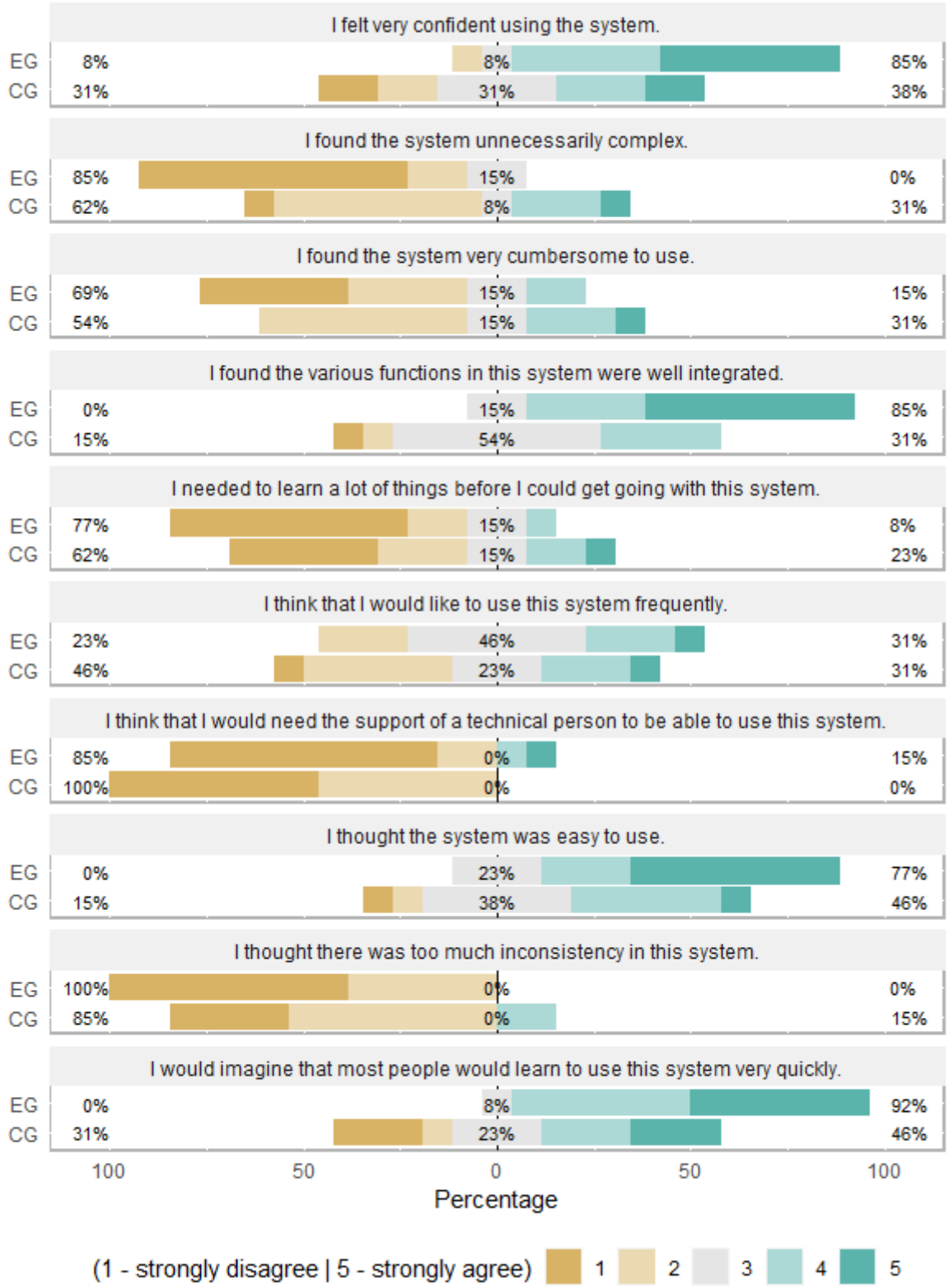


Figure 4.1: Standard System Usablity Scale (SUS) test questions. A clear advantage of the EG working with *CARdiFlow* is visible.

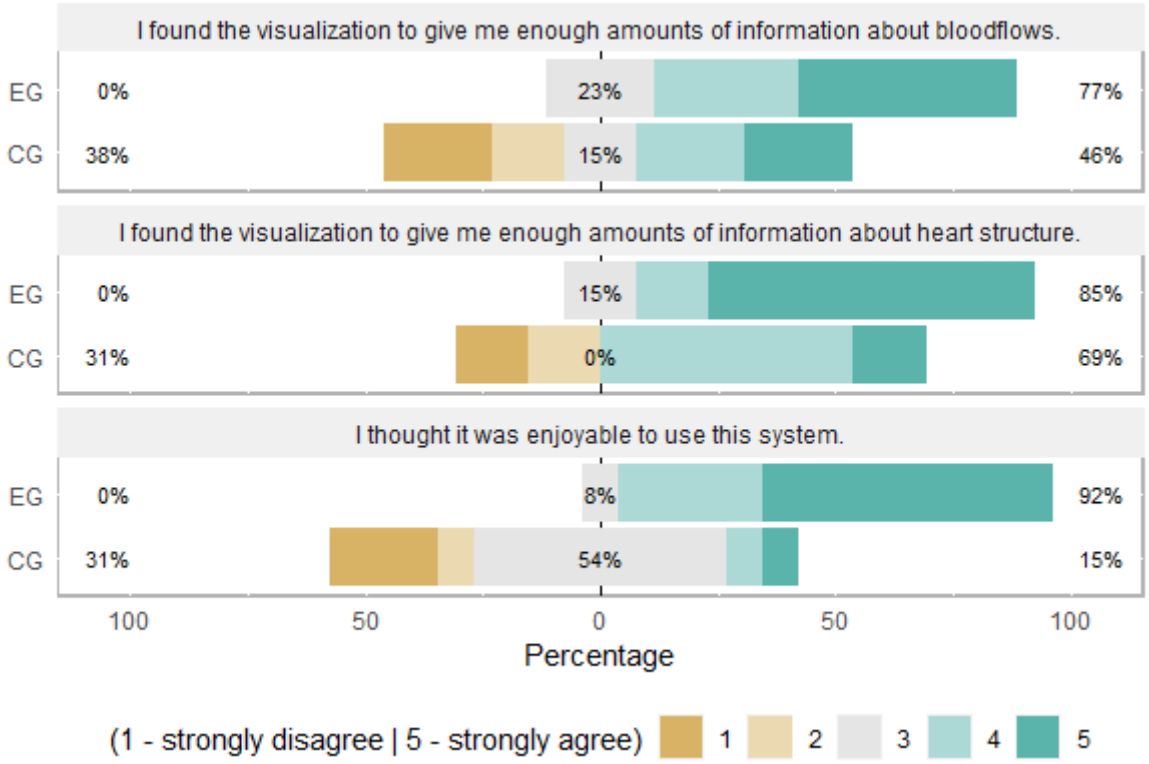


Figure 4.2: Extended System Usablity Scale (SUS) test questions. A clear advantage of the EG working with *CARdiFlow* is visible.

4.2.2 Findings

Statistical Analysis

We used a two tailed t-test to check for scientific significance levels with a p-value of $p < 0.05$.

Q1. Regarding the task related to heart size, there was no significant change between groups, $t(24) = -1.00$, $p = .163643$, despite the control group ($M = 1.00$, $SD = 0.00$) attaining higher scores than the experimentation group ($M = 0.92$, $SD = 0.27$).

Q2. Regarding the task related to blood flow order, there was no significant change between groups, $t(24) = -0.08229$, $p = .467548$, despite the control group ($M = 4.62$, $SD = 2.27$) attaining higher scores than the experimentation group ($M = 4.54$, $SD = 2.31$).

Q3. Regarding the task related to flow area names, there was no significant change between groups, $t(24) = 1.40771$, $p = .086019$, despite the experimentation group ($M = 5.69$, $SD = 1.07$) attaining higher scores than the control group ($M = 4.77$, $SD = 2.01$).

Q4. Regarding the task related to aortic valve placement, the $n = 13$ participants from group EG ($M = 0.69$, $SD = 0.46$) compared to the 13 participants from group CG ($M = 0.31$, $SD = 0.46$) demonstrated significantly better scores, $t(24) = 2.55841$, $p = .017248 < 0.05$.

Q5. Regarding the task related to inferior vena cava placement, the $n = 13$ participants from group EG ($M = 0.85$, $SD = 0.36$) compared to the 13 participants from group CG ($M = 0.31$, $SD = 0.46$) demonstrated significantly better scores, $t(24) = 3.18401$, $p = .001996 < 0.05$.

To summarize, we found a significant improvement concerning anatomical knowledge in users exposed to *CARdiFlow* and a slight improvement in naming cardiac blood flows. No significant change could be seen regarding flow ordering and heart size questioning. All tasks were possible with both approaches, however, as one can see from the SUS scores, there is a strong preference for our AR application. Still, participants recognize that they might need technical support to learn how to use this system.

Interaction Time

Further we wanted to check for a correlation between interaction time and learning effect. Regarding this we found that the $n = 13$ participants from group EG ($M = 15.38$, $SD = 6.03$) minutes compared to the 13 participants from group CG ($M = 10.00$, $SD = 4.39$) minutes were significantly interacting for a longer time, $t(24) = 2.50106$, $p = .009804 < 0.05$.

System Usability Scale (SUS)

To interpret the System Usability Scale we must remember that the values are not percentages and therefore may not be understood as such. Due to a huge research on over 500 SUS-scores by John Brooke ([Bro13]) we may conclude, that values of under 68 are under average, while values over 68 are clearly above average. Therefore, we see the control group to have an underwhelming experience with a mean of 60.19, while the experimentation group did enjoy the system very much with a mean of 80.38. Furthermore the results can now be interpreted by a process similar to grading on a curve (see Fig. 4.3), which sets the traditional learning system on a D and the modern learning system on an A. Interestingly, people are much more likely to propagate a system from grade A on [Bro13].

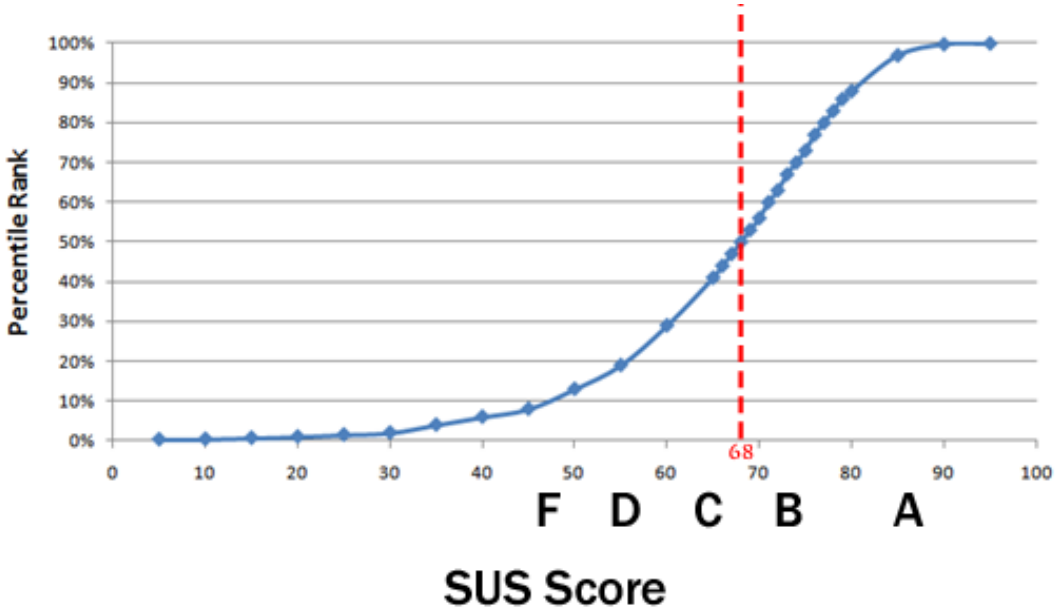


Figure 4.3: A depiction of the SUS score curve grading system developed by Brooke [Bro13]. Note the average (red) is around 68 points.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

We demonstrated that the experimentation group had a significantly better and more efficient learning experience with *CARdiFlow*. The novel approach to display and teach cardiac blood flow in augmented reality lead the users to interact for `50%` longer mean times (`t = 15.38` minutes for EG, comparing to `t = 10.00` minutes for CG) and facilitated the understanding process as well as memorability of anatomical structures.

Even if the improvement of knowledge about the human blood flow was not significantly different, it was still enhanced by quite some amount being $p \approx$ `.468` for ordering and $p \approx$ `.086` for naming. Though the understanding of heart structure greatly increased with values of $p \approx$ `.017` and $p \approx$ `.002`. The participants remembered the locations of the aortic valve and the inferior vena cava better than the control group, which shows the strength of 3D visualizations and the power of interactivity.

Overall the modern, augmented reality-supported learning approach achieved great scores from the System Usability Scale tests. While the control group mean score was only $\overline{x}$ `= 60.38`, the experimentation group scored $\overline{x}$ `= 80.38` units in the SUS test. Evaluating multiple hundreds of SUS scores, showed the mean to be `68` — therefore, the traditional static approach scored underwhelming being graded only *D*, while the proposed approach scored very pleasantly being in the top `10%` of the curve and thereby graded as *A*.

Finally we would want to emphasize that while no clear correlation was identified, it seems only logical that spending more time learning does enhance the learning effect. This means developing systems, which facilitate such behavior (such as interactive AR systems) may result in more effective learning and therefore better results. This can be seen by comparing the overall results of the two groups and their interaction times.

5.2 Future Work

In terms of future directions a lot of work could be put into creating a dynamic personalized beating heart model. This would enable the possibility to create dynamic flows, which correspond to the blood pressure. If new, more performant approaches to fluid visualization were found, or after the computational power hast yet gotten stronger, even realtime flow calculations could take part, making it possible to display the blood flow at different blood temperatures and therefore at different viscosities.

Another area of improvement could be the interaction possibilities included in the application. While it is as of now possible to view the heart model, blood flow and labels for areas, there could be implemented a lot of features making it more of a responsive experience, like zooming, rotating, different levels of see-/cut-through. This would present interested users with a more detailed view, while also engaging not already convinced students to immerse themselves fully and spend more time learning.

Furthermore it would be possible to display additional information on demand. On tapping or zooming certain areas, this would show more information about the selected area, providing in-depth knowledge and maybe even short videos about different parts of focus, like tricuspid valves opening or live footage from an internal heart beating process. This could be even enriched with narrative or storytelling strategies to support better learning.

Lastly, we want to include valuable information collected from our participants. Some experienced frame-drops due to old hardware, which could be solved by enhancing the vertex animation textures to be of smaller disc space. Others wanted more visible guides, like arrows pointing to in- and outflows, or legends describing which color stands for de-/oxygenated blood. Additionally, a guided and more intuitive tutorial would also increase engagement and memorability.

This is the first step towards a future, in which medical students are supported by modern day equipment and technology. We sincerely hope for fellow researchers to implement state of the art mechanisms into education, as it shows time and time again that demand and gain are present.

APPENDIX A

Supplemental Figures

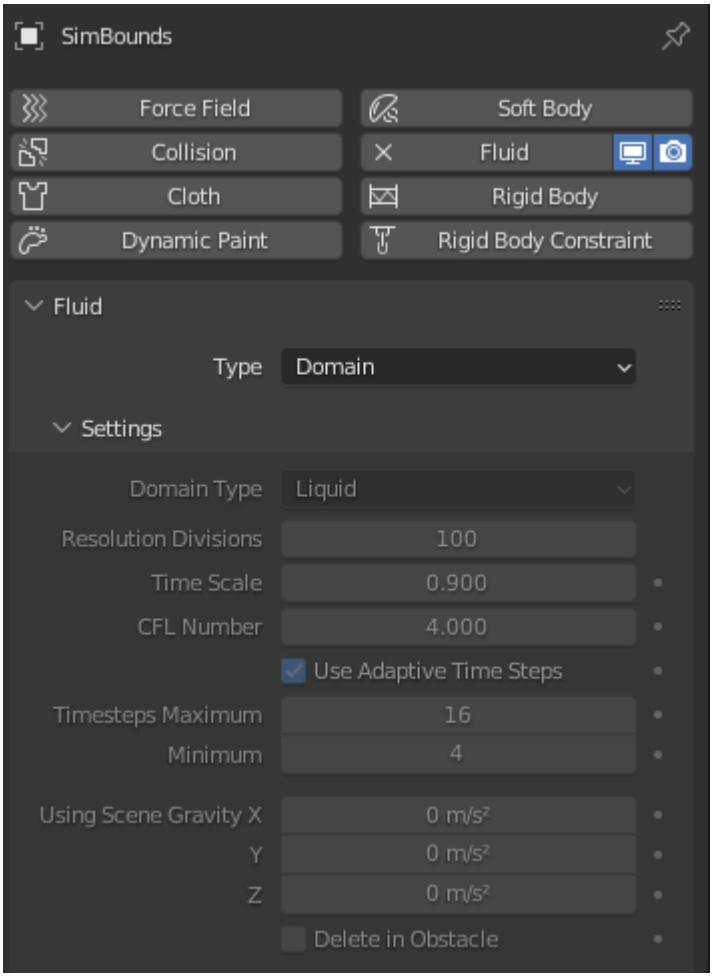


Figure A.1: Fluid domain parameters, including resolution.

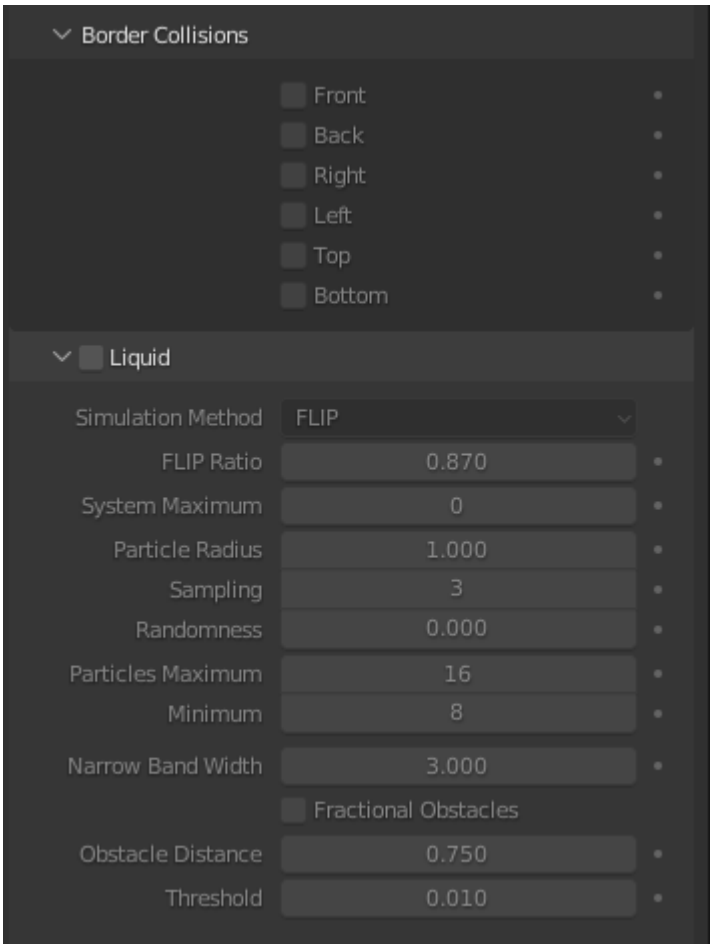


Figure A.2: Fluid domain parameters showing FLIP values.

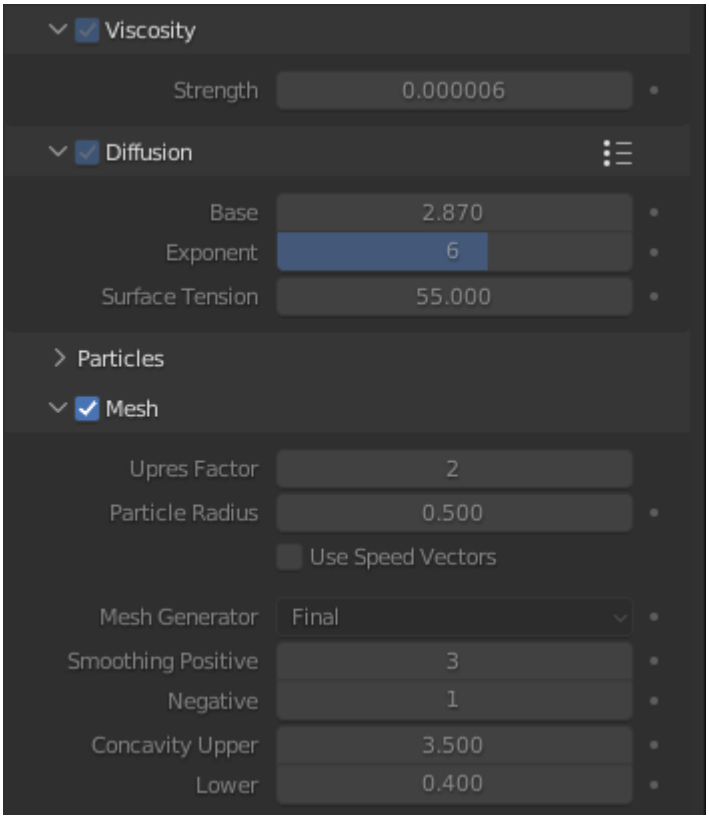


Figure A.3: Fluid domain parameters showing viscosity.

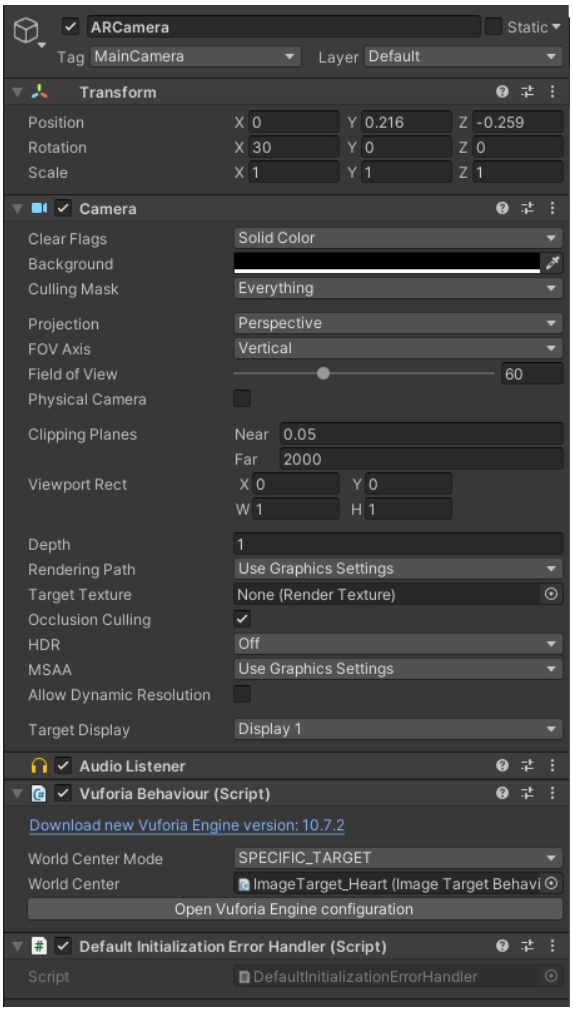


Figure A.4: The `Vuforia` AR Camera with its settings and an especially small near clipping plane value.

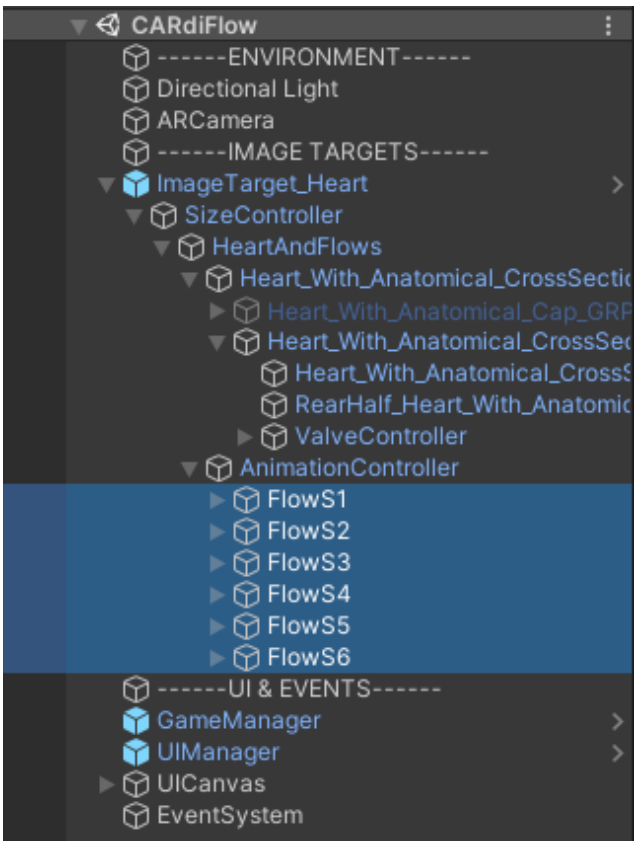


Figure A.5: The scene hierarchy with the 6 cardiac blood flows highlighted.

APPENDIX B

Code

An online repository with the full implementation can be found by following this URL: `https://github.com/Zai-shen/CARdiFlow` [Jan].

Listing B.1: VuforiaFix.cs

```
1 public class VuforiaFix : MonoBehaviour
2 {
3     private void Awake()
4     {
5         try
6         {
7             MethodInfo attachHandler = typeof(Vuforia.VuforiaRuntime).
                  GetMethod("AttachVuforiaToMainCamera", BindingFlags.
                  NonPublic | BindingFlags.Instance);
8
9             Delegate d = Delegate.CreateDelegate(typeof(UnityEngine.Events
                  .UnityAction<Scene, LoadSceneMode>), Vuforia.
                  VuforiaRuntime.Instance, attachHandler);
10            SceneManager.sceneLoaded -= d as UnityEngine.Events.
                  UnityAction<Scene, LoadSceneMode>;
11        }
12        catch (Exception e)
13        {
14            Debug.LogError("Cant remove the AttachVuforiaToMainCamera
                  action. Exception: " + e.Message);
15        }
16        Destroy(this);
17    }
18 }
```

Listing B.2: Part of LabelController.cs

```
1     public void ShowLabels(LabelDisplay lMode)
2     {
3         DisableCurrent();
4         labelDisplay = lMode;
5         switch (lMode)
6         {
7             case LabelDisplay.SPARSE:
8                 ChangeLabelState(labels_sparse, true);
9                 break;
10            case LabelDisplay.MODERATE:
11                ChangeLabelState(labels_moderate, true);
12                break;
13            case LabelDisplay.PRECISE:
14                ChangeLabelState(labels_precise, true);
15                break;
16            case LabelDisplay.NONE:
17            default:
18                break;
19        }
20    }
```

Listing B.3: SizeController.cs

```
1 public class SizeController : UnitySingleton<SizeController>
2 {
3     private const int initialSize = 12;
4
5     [SerializeField]
6     private Vector3 currentSize;
7
8     protected override void Awake()
9     {
10        base.Awake();
11        currentSize = transform.localScale;
12    }
13
14    public void ChangeScale(int newSizeInCm)
15    {
16        currentSize = Vector3.one / initialSize * newSizeInCm;
17        transform.localScale = currentSize;
18    }
19 }
```

Listing B.4: Part of ColorController.cs

```
1     public void SetColors(ColorMode cMode)
2     {
3         this.colorMode = cMode;
4         for (int i = 0; i < Globals.LEFT_HEART.Length; i++)
5         {
6             FlowController.Instance.Flows[Globals.LEFT_HEART[i]].
7                 SetShaderColor(cMode == ColorMode.STANDARD ? blood : red);
8         }
9         for (int i = 0; i < Globals.RIGHT_HEART.Length; i++)
10        {
11            FlowController.Instance.Flows[Globals.RIGHT_HEART[i]].
12                SetShaderColor(cMode == ColorMode.STANDARD ? blood : blue)
                      ;
13        }
14    }
```

List of Figures

List of Tables

Bibliography

[AA17] Murat Akcayir and Gökce Akcayir. Advantages and challenges associated with augmented reality for education: A systematic review of the literature. *Educational Research Review*, 20:1–11, 2017.

[AFr] AFrame.io. A-frame. `https://aframe.io/`. Accessed: 14.9.2021.

[AJM17] Mohamed Ali, Mona Erfani Joorabchi, and Ali Mesbah. Same app, different app stores: A comparative study. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 79–90, 2017.

[And] MJ Anderson. Augmented or virtual: How do you like your reality? `https://www.trekk.com/sites/default/files/inline-images/Reality_Continuum.png`. Accessed: 01.06.2021.

[App] Apple. App store. `https://www.apple.com/at/ios/app-store/`. Accessed: 14.05.2021.

[Ble] Blender. Blender 2.92. `https://www.blender.org/`. Accessed: 15.05.2021.

[Bro95] John Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.

[Bro13] John Brooke. Sus: a retrospective. *Journal of Usability Studies*, 8:29–40, 01 2013.

[cai] caiovm. Use ar camera vuforia core in individual scene not entire project. `https://forum.unity.com/threads/use-ar-camera-vuforia-core-in-individual-scene-not-entire-project.498489`. Accessed: 01.12.2021.

[CGZ] CGZarvandi. Herz & anatomischer querschnitt 3d-modell. `https://www.turbosquid.com/de/3d-models/3d-model-heart-anatomical-cross-section-1424032`. Accessed: 01.05.2021.

[Cra13] Alan B. Craig. *Understanding Augmented Reality: Concepts and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.

[Cur] Curiscope. Virtuali-tee. https://www.curiscope.com/products/virtuali-tee. Accessed: 13.05.2021.

[dH13] Niels de Hoon. Cardiovascular blood flow comparing measurement and simulation. *Computer Graphics Forum*, page 88, 08 2013.

[dHvPJV14] N. de Hoon, R. van Pelt, A. Jalba, and A. Vilanova. 4d mri flow coupled to physics-based fluid simulation for blood-flow visualization. *Computer Graphics Forum*, 33(3):121–130, 2014.

[EG] Inc Epic Games. Unreal engine. https://www.unrealengine.com/en-US. Accessed: 14.11.2021.

[EH97] J. ROSINA E. HRNČÍŘ. Surface tension of blood. *Physiological Research*, 46:319–321, 1997.

[Elb21] Mohamed A. Elblbesy. The refractive index of human blood measured at the visible spectral region by single-fiber reflectance spectroscopy. *AIMS Biophysics*, 8(1):57–65, 2021.

[GLP+20] Alexis A. Gonzalez, Pablo A. Lizana, Sonia Pino, Brant G. Miller, and Cristian Merino. Augmented reality-based learning for the comprehension of cardiac physiology in undergraduate biomedical students. *Advances in Physiology Education*, 44(3):314–322, 2020. PMID: 32568005.

[Gmb] Anton Paar GmbH. Viscosity of whole blood. https://wiki.anton-paar.com/at-de/vollblut/. Accessed: 20.10.2021.

[Gooa] Google. Google glass. https://www.google.com/glass/start/. Accessed: 15.05.2021.

[Goob] Google. Play store. https://play.google.com/store. Accessed: 14.05.2021.

[Gro] Network Working Group. Png (portable network graphics) specification version 1.0. https://www.hjp.at/doc/rfc/rfc2083.html. Accessed: 03.07.2021.

[htt] https://github.com/Gaxil/. Alembic to vat. https://github.com/Gaxil/Unity-AlembicToVAT. Accessed: 12.08.2021.

[Inca] Apple Inc. Arkit. https://developer.apple.com/augmented-reality/arkit/. Accessed: 12.8.2021.

[Incb] RheoSense Inc. Viscosity units. https://www.rheosense.com/basics/viscosity-units. Accessed: 21.10.2021.

[Jan] Miran Jank. Cardiflow github repository. https://github.com/Zai-shen/CARdiFlow. Eingesehen am 18.10.2022.

[KBvP+17] Benjamin Köhler, Silvia Born, Roy F. P. van Pelt, Anja Hennemuth, Uta Preim, and Bernhard Preim. A survey of cardiac 4d pc-mri data processing. *Computer Graphics Forum*, 36(6):5–35, 2017.

[Lee19] et al. Lee, Geon-hee. A study on the performance comparison of 3d file formats on the web. *International Journal of Advanced Smart Convergence*, 8(1), 2019.

[Leh] Shawn Lehner. Armaker. https://shawnlehner.github.io/ARMaker/. Accessed: 05.05.2021.

[LLCa] Google LLC. Android studio. https://developer.android.com/studio. Accessed: 14.11.2021.

[LLCb] Google LLC. Ar core. https://developers.google.com/ar. Accessed: 12.8.2021.

[Lum] LumenLearning.com. Systolic and diastolic pressures. https://courses.lumenlearning.com/suny-ap2/chapter/blood-flow-blood-pressure-and-resistance-no-content/. Accessed: 13.03.2022.

[Mic] Microsoft. Hololens 2. https://www.microsoft.com/hololens. Accessed: 15.05.2021.

[MSRS17] Christian Moro, Zane Stromberga, Athanasios Raikos, and Allan Stirling. The effectiveness of virtual and augmented reality in health sciences and medical anatomy. *Anatomical Sciences Education*, 10(6):549–559, 2017.

[MTUK94] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351, 01 1994.

[N.Aa] N.A. N.a. https://open3dmodel.com/wp-content/uploads/2020/05/Body-mesh-Sheets.png. Accessed: 14.05.2022.

[N.Ab] N.A. N.a. https://miro.medium.com/max/1038/0*ma0MfhAK1rgfcF2y. Accessed: 14.05.2022.

[ØL15] Tina Øvad and Lars Bo Larsen. The prevalence of ux design in agile development processes in industry. *2015 Agile Conference*, pages 40–49, 2015.

[PB13] Bernhard Preim and C.P. Botha. Visual computing for medicine: Theory, algorithms, and applications: Second edition. *Visual Computing for Medicine: Theory, Algorithms, and Applications: Second Edition*, pages 1–812, 11 2013.

[PTC] PTC. Vuforia engine 9.8. https://developer.vuforia.com/. Accessed: 15.05.2021.

[PW91] Frits Post and Theo Walsum. Fluid flow visualization. In *Fluid Flow Visualization*, pages 1–40, 01 1991.

[Rai] Raideos. Heart. https://clara.io/view/bef08af4-bf65-4f9f-ab96-e3c8a42fdd3b. Accessed: 01.05.2021.

[SAhY15] Nor Saidin, Noor Abd halim, and Noraffandy Yahaya. A review of research on augmented reality in education: Advantages and applications. *International Education Studies*, 8:8, 06 2015.

[SOJS+17] Patrick Saalfeld, Steffen Oeltze-Jafra, Sylvia Saalfeld, Uta Preim, Oliver Beuing, and Bernhard Preim. Sketching and Annotating Vascular Structures to Support Medical Teaching, Treatment Planning and Patient Education. In Stefan Bruckner and Timo Ropinski, editors, *EG 2017 - Dirk Bartz Prize*. The Eurographics Association, 2017.

[Son] Lucasfilm & SonyPictures. Alembic. https://www.alembic.io/. Accessed: 15.05.2021.

[SSJ16] Lamia Soussi, Zeena Spijkerman, and Slinger Jansen. A case study of the health of an augmented reality software ecosystem: Vuforia. In Andrey Maglyas and Anna-Lena Lamprecht, editors, *Software Business*, pages 145–152, Cham, 2016. Springer International Publishing.

[Sut] Fia Sutton. Does choosing a gltf over an obj file format matter? https://fiasutton.medium.com/does-choosing-a-gltf-over-an-obj-file-format-matter-42ad5a1794bc Accessed: 22.04.2022.

[Teca] Unity Technologies. Ar foundation. https://unity.com/de/unity/features/arfoundation. Accessed: 12.8.2021.

[Tecb] Unity Technologies. Render pipelines. https://docs.unity3d.com/Manual/render-pipelines.html. Accessed: 17.04.2022.

[Tecc] Unity Technologies. Unity. https://unity.com/. Accessed: 15.05.2021.

[Tecd] Unity Technologies. Unity mars. https://unity.com/de/products/unity-mars. Accessed: 12.8.2021.

[Tece] Unity Technologies. Xr interaction toolkit. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html. Accessed: 12.8.2021.

[Tur] TurboSquid. 3d-modelle für profis. https://www.turbosquid.com/de/. Accessed: 01.05.2021.

[Ver] Simon Verstraete. Vertex animation textures in unreal. https://www.sidefx.com/tutorials/vertex-animation-textures-for-unreal/. Accessed: 14.05.2022.

[Vufa] Vuforia. Best practices for designing and developing image-based targets. https://library.vuforia.com/objects/best-practices-designing-and-developing-image-based-targets. Accessed: 06.06.2021.

[Vufb] Vuforia. Image targets optimization techniques. https://library.vuforia.com/objects/image-targets-optimization-techniques. Accessed: 06.06.2021.

[ZHMZ14] Egui Zhu, Arash Hadadgar, Italo Masiello, and Nabil Zary. Augmented reality in healthcare education: an integrative review. *PeerJ*, 2:e469, July 2014.