HTL Leonding
next level

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING

Animotion - Animation through Motion

DIPLOMA THESIS

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Submitted by:
Romeo Bhuiyan
Christoph Lasinger

Supervisor:
FH-Prof. Dipl.-Ing. Dr. Erik Sonnleitner

Leonding, April 2023

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such. This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 2023 Romeo Bhuiyan & Christoph lasinger

Abstract

Animotion is a combination of the two words animation and motion, cramming the general idea behind it into as little information as possible. This general idea being a camera, for example that of a laptop, a phone or an external one used in a computer setup, recording face and body gestures of the user and translating them onto a virtual reality model (VRM), thereby controlling it. This is done by using an artificial intelligence that calculates tracked gestures and puts them into a canvas that displays the default pose of the VRM and additional user input recorded by the camera. The aforementioned model can be selected by the user out of an assortment of three possible choices on the main page of the website. The moving VRM can be recorded, either by recording the browser window or the entire screen, and for example posted on social media for entertainment purposes. Another possible application would be using Animotion as a way of recording oneself for one's own livestream ("V-Tubing"), where, for a variety of reasons, one chooses to not represent oneself with one's own body but a virtual ("fictional") one instead.

The frontend, i.e., the website was mainly implemented using a combination of the JavaScript framework Next.js and Sass, a stylesheet language similar to CSS, or more accurately, an extension of it, used for the general design. For the web application MediaPipe, an open-source, cross-platform framework was used to build machine learning solutions for streaming media, and Holistic, a gesture analysis and control library of MediaPipe that is mainly used for augmented reality effects, were chosen in order to depict the VRM together with three.js, a JavaScript library used to create and display 3D computer graphics in a web browser.

Zusammenfassung

Animotion ist eine Kombination aus den beiden Worten „Animation“ und „Motion“, wobei die generelle Idee dahinter in so wenig Information wie möglich zusammengefasst wird. Diese Idee ist, dass eine Kamera, zum Beispiel die eines Laptops, eines Handys oder eine externe Kamera in einem Computer-Setup, die Mimik und Gestik des Benutzers aufzeichnet und auf ein „Virtual Reality Model (VRM)“ überträgt, um es dadurch zu steuern. Dies wird durch eine künstliche Intelligenz erreicht, die verfolgte Gesten berechnet und sie auf eine Leinwand überträgt, die die Standardpose des VRM und zusätzliche, aufgenommene Benutzerbewegungen darstellt. Das eben erwähnte Modell kann aus einer Auswahl von drei möglichen Optionen auf der Hauptseite der Website selektiert werden. Das bewegende VRM kann aufgezeichnet werden, indem entweder das Browserfenster oder der gesamte Bildschirm aufgenommen wird, und zum Beispiel auf sozialen Medien zu Unterhaltungszwecken gepostet werden. Eine weitere mögliche Anwendung wäre die Verwendung von Animation zur Aufzeichnung von einem selbst für eine eigene Liveübertragung („V-Tubing“), bei der man aus verschiedenen Gründen sich selbst nicht mit dem eigenen Körper, sondern einem virtuellen („fiktionalen“) Körper darstellen möchte.

Das Frontend, also die Website, wurde hauptsächlich mithilfe einer Kombination aus dem JavaScript-Framework Next.js und Sass, einer Stylesheet-Sprache ähnlich wie CSS, oder genauer gesagt eine Erweiterung, die für das allgemeine Design verwendet wird, implementiert. Für die Web-Anwendung wurden MediaPipe, ein open-source, plattformübergreifendes Framework, das zum Erstellen von Machine Learning-Lösungen für Übertragungsmedien verwendet wird, und Holistic, eine Gestenanalyse- und Steuerungsbibliothek von MediaPipe, die hauptsächlich für Erweiterte Realitätseffekte verwendet wird, ausgewählt, um das VRM zusammen mit three.js, einer JavaScript-Bibliothek, die verwendet wird, um 3D-Computergraphiken in einem Webbrowser anzuzeigen, darzustellen.

Contents

1 Introduction

2 Artificial intelligence

2.1 What is artificial intelligence?

The concept of AI has a long history that dates back to ancient times, when people first tried to build machines that could mimic human abilities. In the modern era, the term *AI* was first coined in 1956 by computer scientist John McCarthy, who defined it as *the science and engineering of making intelligent machines* [1].

Today, these machines are designed to be able to perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and understanding natural language. AI can be applied to a wide range of fields, from healthcare and finance to education and transportation, with the goal of making systems more efficient and effective. AI can be classified into two broad categories: *narrow or weak AI*, which is designed to perform a specific task, and *general or strong AI*, which has the ability to perform any intellectual task that a human being can perform. [1]

2.2 Philosophy of artificial intelligence

Can a machine think? Thinking seems to be one of, if not the most important part of being human, as seen in René Descartes' first principle of philosophy "Cogito, ergo sum" ("I think, therefore I am"). So it goes to reason that, if we, as homo sapiens, were to see us superior to other animals on earth, it would be our intelligence, our conscious thinking, that would seem to differentiate us, make us special. Assuming this line of reasoning to be true, then it would only seem logical to think that we, as humans, are superior to machines in that regard as well. But what if that were to be taken away?

Throughout the course of history humans have practically been obsessed with self-imitation, be it wall-paintings, statues, portraits, photos, films or today's machines. The

[1] johnmcCarthy:1956

earliest example of artificial humans may be Greek gods, or rather Greek mythology as a whole, including for example architect and craftsman Daedalus, who created statues and machines that were sometimes impossible to distinguish from real human beings. Ancient Egypt also featured statues of gods behaving like humans and even though they were being controlled not by a god, but through complex mechanisms, such as quicksilver or hydraulics, or even simple puppeteering strings, the Egyptians still feared and revered them. They saw it not as blasphemy but as gods acting through guided souls, even though they might be that of a master and puppet.

However, not everyone would view such creations as positive as the Greeks and Egyptians. As even before their time a different culture, or to be more accurate religion had rules regarding artificial humans, the so called ten commandments, of which the second one states “You shall not make for yourself a carved image, or any likeness of anything that is in heaven above, or that is in the earth beneath, or that is in the water under the earth. You shall not bow down to them or serve them, for I the LORD your God am a jealous God ...". Of course, any religious text is open to interpretation, but it seems a rather popular one was that of Hermes Trismegistor, purported author of the Hermetica, a series of ancient texts that lay the basis of philosophical systems known as Hermetecicism, who stated that man has created statues, infused with the souls of demons and angels through holy rituals.

The purpose of mentioning these two views regarding artificial humans is that they are essentially the two fundamental views of the western world regarding “thinking machines”, or artificial intelligence, as well. This can be seen in statements made by both critics and proponents of AI.

2.3 The future of artificial intelligence

The future of AI is difficult to predict with certainty, but it is likely that AI will continue to advance and become increasingly integrated into our daily lives. AI has the potential to revolutionize many industries, from healthcare and transportation to education and finance. It could also have a major impact on the job market, with some jobs being automated by AI and other jobs being created to support the technology. However, it is important to consider the potential drawbacks of AI and ensure that it is developed and used ethically. Overall, the future of AI seems promising, but it is important to approach it with caution and consideration. As the future of AI is very likely to lie somewhere between a very positive one (extreme optimism) and a very negative one (extreme pessimism), it makes sense to analyze those two specifically.

An extremely pessimistic future of AI in the world would involve the technology being used in ways that are harmful to humanity. In this scenario, AI could be used to create weapons of mass destruction, or to control and manipulate people for nefarious purposes. It could also be used to create a surveillance state, where people's every move is monitored and tracked. Additionally, the widespread use of AI could lead to widespread job loss and economic instability, as many jobs are automated and replaced by machines. In a worst-case scenario, the development of AI could even lead to a global conflict over control of the technology.

An extremely optimistic future would entail an AI with highly advanced and integrated technology that is able to solve many of the world's most pressing problems. In this scenario, AI would be used to improve healthcare, reduce poverty and inequality, and address climate change. It would also be used to make transportation faster, safer, and more efficient, and to improve education by providing personalized learning experiences for students. Moreover, AI could be used to help us better understand and protect the natural world, and to explore the universe. Overall, this extremely optimistic future of AI would involve the technology being used to enhance and improve human life in countless ways.

2.4 The functional concept of neural networks

A neural network is a type of machine learning algorithm modeled after the structure and function of a human brain (neural linking). It is composed of many interconnected processing nodes, called neurons, which work together to process information.

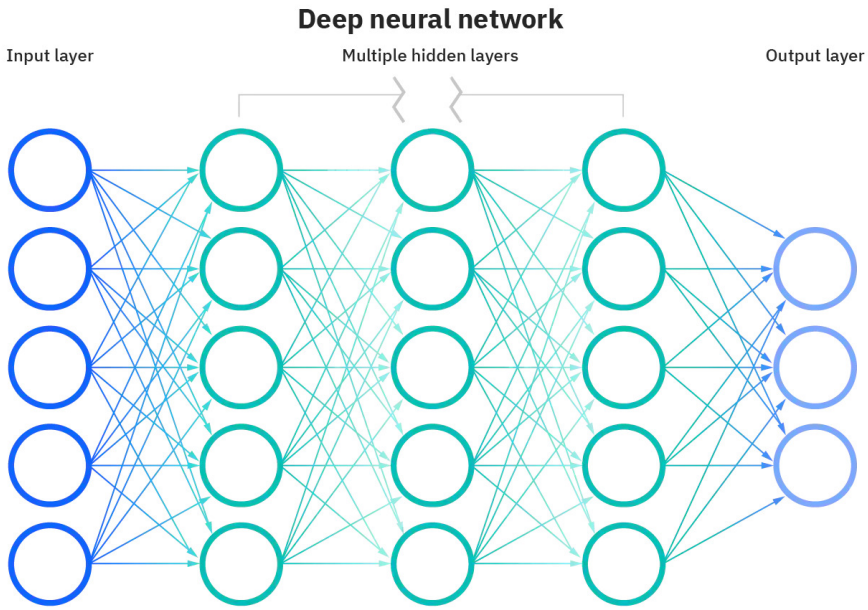

Figure 1: Complex neural network
[2]

As shown in figure 1, each neuron receives input from other neurons, processes that information, and produces an output. This output is then passed on to other neurons in the next layer of the network. In this way, information is passed through the network, from the input layer to the output layer, allowing the neural network to learn and make predictions based on the data it is given. [3]

The specific details of how a neural network works can vary depending on its architecture and the type of problem it is being used to solve. But in general, a neural network is able to learn from data by adjusting the strength of the connections between its neurons, these connections being called weights, based on the input it receives. Over time, the network is able to improve its predictions by adjusting these weights in a way that minimizes errors between the network's output and the correct output.

2.4.1 Solving methods

There are three basic methods for solving problems: search-, knowledge-, and algorithmic methods. Every method involves searching through a space of possible solutions whilst optimizing a pre-defined evaluation function, that can lead to the following *Combinatorial explosion* as shown below in the figure 2. The methods can range from simplex hill climbing via alpha-beta pruning techniques to knowledge *chunking*.

Example: the chess endgame of king and three pawns versus king and three pawns requires an explicit table of half a billion moves and a run-time of ten quadrillion years to evaluate all possibilities. The success of the CMU *chunker* program which uses chess domain knowledge chunks is that it reduces this run-time to about one minute.

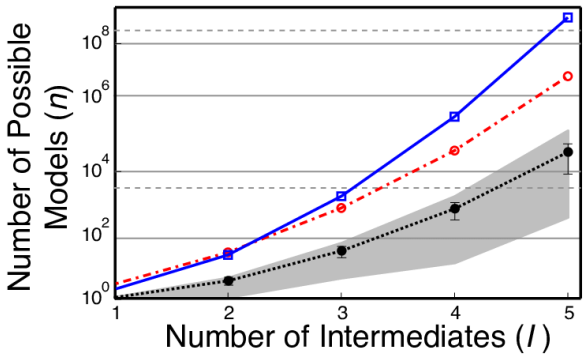

Figure 2: Combinatorial explosion

2.5 The science and technology behind face tracking

Face tracking is a technology that allows a computer or device to identify and monitor the movements of a person's face in real time. This is typically done using a combination of computer vision algorithms and specialized hardware, such as a camera or depth sensor. [4]

To track a face, the system first detects the face in the video feed from the camera or depth sensor. This is typically done using a machine learning algorithm trained to recognize faces in images. Once the face has been detected, the system then uses various techniques to track the movements of the face, such as tracking the position of

key facial features (such as the eyes and mouth) over time, as seen blow in the figure 3. This allows the system to accurately follow the face as it moves within the frame, even if it turns or changes orientation.

The resulting data can be used for a variety of purposes, such as enabling facial recognition (to identify who the person is), animating virtual characters, or controlling a user interface.

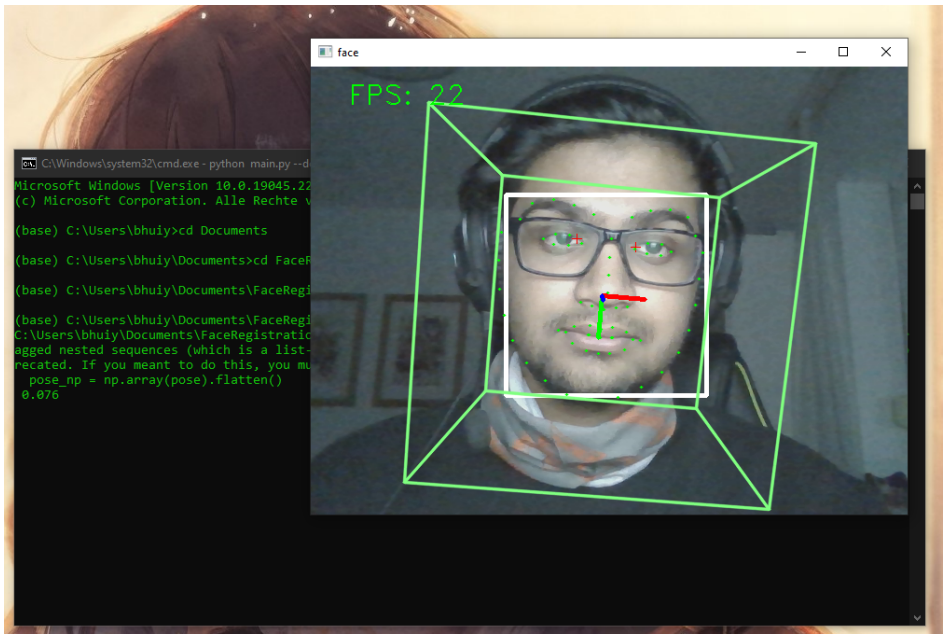

Figure 3: Face tracking experiment done by Romeo Bhuiyan

2.6 The science of body tracking

Body tracking is the process of using technology to track the movement of a person's body. This is typically done using sensors or cameras that capture the movement of the body and then use algorithms to interpret that movement and translate it into digital data that can be used for various purposes.

To track a body, the algorithm first detects the presence of a body in the video or data. It then uses various techniques to identify specific features of the body, such as the limbs, torso, or other distinctive features. This allows the algorithm to track the body as it moves over time.

In addition to tracking the location of the body, some algorithms can also track other features, such as body movements or gestures. This allows them to be used for a variety of applications, such as video surveillance, virtual reality, gaming, human-computer interaction, and fitness tracking.

2.7 Comparison of machine learning frameworks

In this case face tracking was needed in order to get a virtual model animated. The two libraries MediaPipe and TensorFlow.js were tested in python to decide which one is better suited for our purpose. It is difficult to say which approach is more suitable for face, hand, and body tracking in the browser, as it will depend on the specific requirements and constraints of one's project. Both MediaPipe and TensorFlow.js are powerful tools that can be used to perform these tracking tasks, but they have different strengths and limitations.

MediaPipe is a library developed by Google that is specifically designed for real-time multimedia processing tasks, such as hand and face tracking. It is written in C++ with some Python bindings and can be used to build cross-platform pipelines for performing various computer vision and machine learning tasks. MediaPipe is optimized for low-latency, real-time processing and is used to build applications that run on a variety of platforms, including the browser. [5]

In contrast, **TensorFlow.js** is a JavaScript library for training and deploying machine learning models in the browser. It is based on the TensorFlow library, a popular machine learning library developed by Google. TensorFlow.js allows you to build and train machine learning models using JavaScript, and it can be used to perform a variety of tasks, including image and text classification, time series forecasting, and natural language processing. [6]

In **Conclusion**, both MediaPipe and TensorFlow.js are able to handle face and hand tracking in the browser, but personal experiments were conducted in order to find out that they have different trade-offs and may be better suited for different types of projects. MediaPipe is optimized for real-time processing and is good for building applications that require low latency, such as augmented reality or interactive applications. However, TensorFlow.js is a general-purpose machine learning library that is suited for building

machine learning models and deploying them in the browser, but it may not be as efficient for real-time processing as MediaPipe. Thinking that in the future more content will likely be added to the project, we have decided to use MediaPipe as our real-time multimedia library solution.

2.8 Performance

Performance is important in the field of AI for a number of reasons. Firstly, the goal of AI is to mimic human intelligence and decision-making, for which performance is a key factor in determining how well a machine is able to do this. In order for AI to be useful and effective, it must be able to perform tasks at a level that is comparable to or better than a human. [7]

Performance is also essential due to its determination of speed and efficiency of an AI system. In many cases, the ability of AI to quickly and accurately process large amounts of data and make decisions based on that data can be a key factor in its success. For example, in the field of finance, a high-performing AI system can help traders make faster, more informed decisions, which can lead to better investment returns.

Additionally, it can affect the cost and feasibility of implementing an AI system. If an AI system is not able to perform well, it may be too expensive or too unreliable to be used in practice. As a result, the performance of AI systems is a critical factor that must be considered in the development and deployment of these technologies.

2.8.1 Determination of performance

There are a few different factors that can be used to determine whether an AI system is performing well in terms of speed and efficiency as seen below in the figure 4. These can include the following:

Throughput is a measure of the amount of data that an AI system is able to process in a given amount of time. It is often used as a metric to evaluate the performance of AI systems, particularly those that are designed to handle large volumes of data or to perform real-time processing tasks.
In general, a system with high throughput is able to process data quickly and efficiently,

while a system with low throughput may be slower and less efficient. The specific throughput requirements for an AI system will depend on the specific task it is designed to perform and the constraints of the environment in which it is operating.
For example, a self-driving car may require a high throughput AI system to process data from sensors and make decisions in real-time, while a machine learning model used for image classification may require a lower throughput due to the relatively lower volume of data being processed.

Latency is a measure of the time it takes for an AI system to respond to a request or input. It is often used as a metric to evaluate the performance of AI systems, particularly those that are designed to perform real-time tasks or to provide a timely response to user inputs.
In general, a system with low latency is able to provide a response quickly and efficiently, while a system with high latency may be slower and less responsive. The specific latency requirements for an AI system will depend on the specific task it is designed to perform and the constraints of the environment in which it is operating.
For example, a real-time translation system may require a low latency AI system to provide fast translations as the user speaks, while a machine learning model used for image classification may have a higher latency due to the time required to process and analyze the image data. Overall, low latency is important for providing a smooth and seamless user experience in interactive and real-time applications.

Accuracy is a measure of the ability of an AI system to produce correct results. It is often used as a metric to evaluate the performance of AI systems, particularly those that are designed to make predictions or decisions based on data.
In general, a system with high accuracy is able to produce correct results consistently, while a system with low accuracy may be prone to errors and produce incorrect results. The specific accuracy requirements for an AI system will depend on the specific task it is designed to perform and the consequences of making an incorrect decision or prediction.
For example, a machine learning model used for medical diagnosis may require a high accuracy to avoid misdiagnosis or harm to patients, while a machine learning model used for recommending products to online shoppers may be able to tolerate a lower accuracy due to the relatively lower consequences of making a incorrect recommendation. Overall, accuracy is important for ensuring that an AI system is able to produce reliable and trustworthy results.

Resource utilization is a measure of the amount of computing power, memory, and other resources that an AI system uses to perform a task. It is often used as a metric to evaluate the performance of AI systems, particularly those that are designed to run on resource-constrained devices or in environments with limited resources.
In general, a system that is efficient in its use of resources is able to perform well with a minimal amount of resources, while a system that is inefficient may require a larger amount of resources to perform the same task. The specific resource utilization requirements for an AI system will depend on the specific task it is designed to perform and the constraints of the environment in which it is operating.
For example, a machine learning model used on a smartphone may need to be efficient in its use of resources to avoid draining the battery or slowing down the device, while a machine learning model running on a server with ample resources may be able to tolerate a higher resource utilization. Overall, resource utilization is important for ensuring that an AI system is able to perform well within the constraints of its environment.

Overall, a well-performing AI system is one that is able to handle a large volume of data quickly, provide a timely response, produce accurate results, and use resources efficiently.

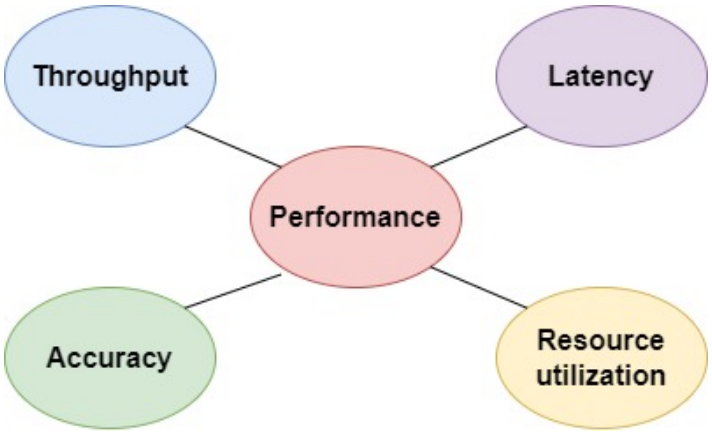

Figure 4: Key factors for determining the performance of an AI system

3 System architecture

3.1 Three.js

Three.js is a JavaScript library that allows for the creation of interactive 3D graphics in the web browser. It is designed to work seamlessly with WebGL, which is a web-based 3D graphics API that allows developers to access the hardware-accelerated graphics capabilities of a user's computer. Three.js makes it possible to create rich 3D experiences with JavaScript and HTML, without the need for any plugins or downloads. With Three.js, developers can create a wide variety of 3D scenes and animations, including but not limited to: realistic environments with dynamic lighting and shadows, complex 3D models and animations, interactive 3D games and simulations, and virtual reality (VR) and augmented reality (AR) experiences.

The library provides a large number of powerful features and abstractions that simplify the process of creating 3D graphics. It includes a number of built-in shapes, such as cubes, spheres, and cylinders, as well as more complex shapes that can be constructed from the vertices and faces of custom 3D models. Three.js also includes a number of materials that allow for realistic shading and texturing of 3D objects, as well as lights and cameras that can be positioned and manipulated to achieve the desired lighting and perspective effects. Another important aspect of Three.js is its support for keyframe animations. This allows developers to animate objects over time, creating smooth transitions and movements. The library also provides support for physics simulations, making it possible to create complex and interactive 3D experiences that respond to user input.

3.1.1 Three-vrm.js

Three-vrm.js is a JavaScript library that provides support for rendering and manipulating VRM (Virtual Reality Model) characters in web-based environments. It is

built on top of Three.js, a widely-used library for creating 3D graphics and animations in web browsers. Three-vrm.js offers a set of tools and APIs for working with VRM models, allowing developers to easily import, manipulate and animate VRM characters in their web-based projects. This includes support for importing VRM files, as well as manipulating and animating individual bones, rigging, morph targets, and materials of the characters as shown below in the figure 5. In addition to basic VRM functionality, three-vrm.js also includes a set of VRM-specific features and extensions. For example, it provides support for VRM's humanoids, which define the structure of a VRM character and allow for easy animation of the character. It also includes support for VRM's blend shapes, which are used to manipulate the shape of the character's face and expressions.

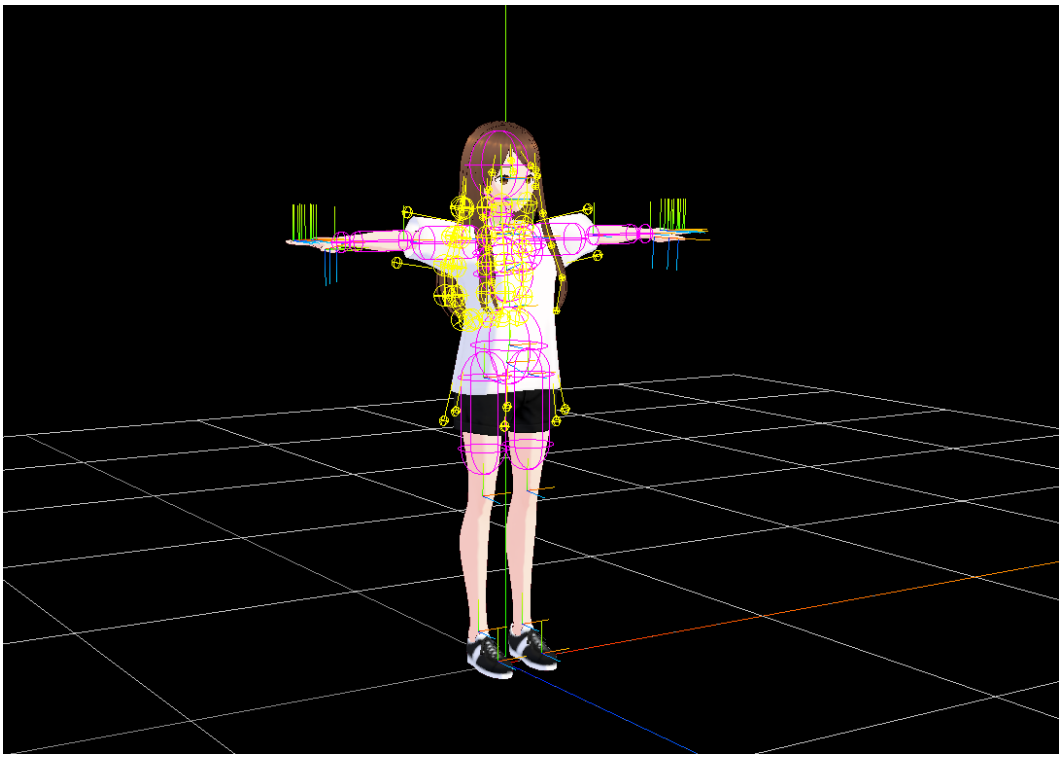

Figure 5: Three-vrm.js debug mode

3.2 Comparison of 3D rendering technologies

In order to display a virtual model on a canvas, a 3D rendering technology was required. Two libraries, GLTFLoader and WebGL, were evaluated using JavaScript to determine

the best fit for the task at hand. The choice between the two ultimately depends on the specific needs and limitations of the project. Both GLTFLoader and WebGL are effective tools for rendering, but each have their own unique strengths and weaknesses.

3.2.1 WebGL

WebGL is designed to work seamlessly with other web technologies such as HTML, CSS, and JavaScript, making it easy to integrate 3D graphics into web pages. It allows developers to create a wide range of interactive 3D applications and visualizations, including games, scientific simulations, data visualizations, and more. One of the key features of WebGL is its ability to take full advantage of the GPU (graphics processing unit) on the user's device. This allows WebGL applications to run smoothly and efficiently, even on devices with limited resources. Additionally, WebGL provides a high level of compatibility across different browsers and devices, making it a widely accessible technology. WebGL is supported by most modern web browsers, including Chrome, Firefox, Safari, and Edge. This means that developers can create WebGL applications that can be easily accessed by users on a wide range of devices and platforms.

3.2.2 GLTFLoader

GLTFLoader is a JavaScript library that allows developers to load and parse 3D models in the GLTF (GL Transmission Format) file format. GLTF is a widely adopted file format for 3D models and is supported by many 3D modeling software, including Blender and SketchUp. GLTFLoader provides a simple and easy-to-use API for loading and parsing GLTF models, which can then be displayed using WebGL. The library takes care of all the complex tasks involved in loading and parsing the model data, such as handling binary buffers, parsing JSON data and creating the geometry and materials for the model. GLTFLoader also provides support for advanced features such as animations, skinning and morph targets, which are typically found in more complex 3D models. This allows developers to create more advanced and interactive 3D applications and visualizations. One of the key benefits of using GLTFLoader is that it greatly simplifies the process of loading and displaying 3D models on the web.

By using a standardized file format and a dedicated library, developers can focus on creating the logic and functionality of their application, rather than spending time on complex parsing and loading tasks. Additionally, GLTF is a highly efficient format, which means that it can be loaded faster, and also the files are smaller in size, making it a great choice for loading 3D models on the web.

3.2.3 Conclusion

n conclusion, GLTFLoader is a powerful tool for creating 3D graphics for the web that offers several advantages over WebGL. One of the main benefits of using GLTFLoader is its ability to handle the loading and parsing of GLTF (GL Transmission Format) files, which are a widely-used format for 3D models. This eliminates the need for developers to write their own code to handle file loading and parsing, saving time and effort. Additionally, GLTFLoader provides a more streamlined and convenient way to work with 3D models, as it allows developers to easily import and use pre-existing models rather than having to create them from scratch. Furthermore, GLTFLoader also supports various features such as animations, cameras, lights and many more which can be easily integrated with web pages. Therefore, it is more efficient and convenient for the purpose of Animotion to work with 3D models on the web, GLTFLoader is the best choice as seen below in the figure. 6

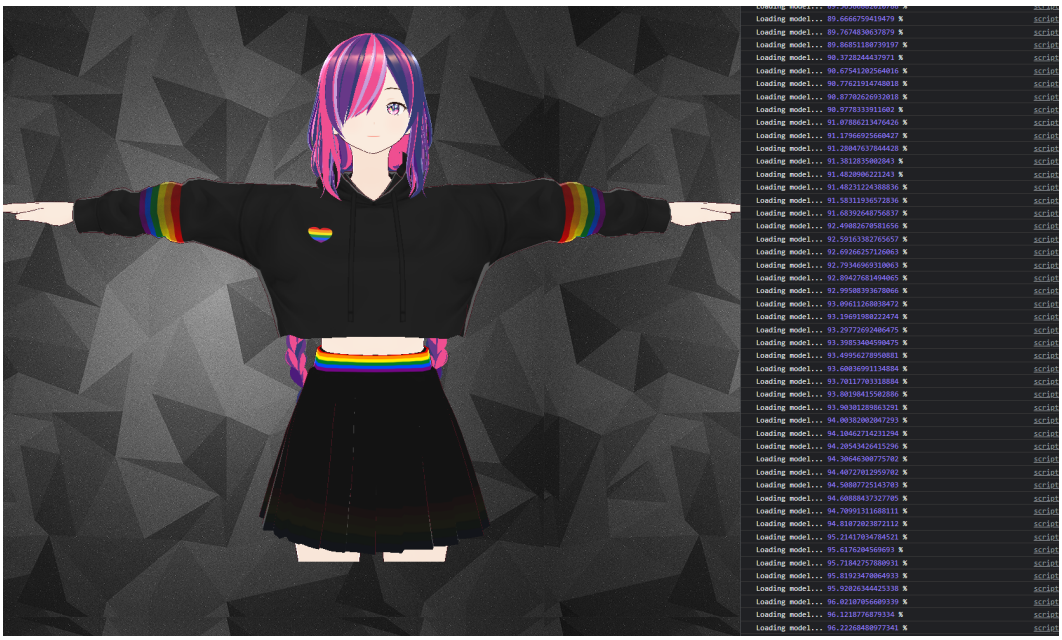

Figure 6: Loading the model with the help of GLTFLoader

3.3 Calculation of the virtual reality model

Controlling a VRM on a website involves a complex process that involves capturing the user's inputs, updating the VRMs position and orientation in real-time, and rendering the updated model for display on the user's device. The first step in this process is for the user to interact with the VR model using a device such as a VR headset or a mouse and keyboard. The inputs from the user are then captured by the website and sent to the server. The server then uses these inputs to update the VR model's position and orientation in real-time.

Once the VR model has been updated, it is then rendered, and the new frame is sent back to the user's device for display. This process repeats as the user continues to interact with the VR model. To achieve this, various technologies are needed such as GLTFLoader[8] from the THREE[9] libraries for rendering the VR model in the browser, WebSockets for real-time communication between the client and server, and a physics engine to simulate the movement and interactions of the VR model.

3.3.1 Blendshape

Blendshape is a technique used in 3D animation and computer graphics to create a smooth transition between different shapes or expressions of a 3D model. It works by creating a set of *target* shapes for a 3D model, each representing a different expression or shape, and then using a set of weights or *blend* values to interpolate between these target shapes, creating a smooth transition between them. This allows animators to create a wide range of expressive characters with a limited number of 3D models. Blendshapes are commonly used in animation, film and game industry.

3.3.2 Lerp

The Lerp function, short for Linear Interpolation, is a mathematical function that is used to smoothly blend between two values. The function takes three arguments: a start value, an end value, and a weight. The weight is a value between 0 and 1 that represents the proportion of the blend between the start and end values. For example, if the weight is set to 0, the function will return the start value, and if the weight is set to 1, the function will return the end value. If the weight is set to 0.5, the function will return the midpoint between the start and end values. By adjusting the weight, the Lerp function can be used to smoothly transition between any two values over time. In the context of computer graphics, Lerp function is widely used for interpolating values such as position, rotation, and color, to make the animation smooth. It's also used in physics simulations, game development, and other fields where the smooth transition of values is needed.

3.3.3 Determining mouth expressions

Listing 1: Shape of mouth

```
1       for each shape in ["I", "A", "E", "O", "U"]
2           Blendshape.setValue(PresetName[shape],
3           lerp(riggedFace.mouth.shape[shape],
4           Blendshape.getValue(PresetName[shape]), 0.5))
```

This pseudocode 1 is a loop that iterates over an array containing the shapes `A, E, I, O, U`. The purpose of the loop is to blend the shape of a rigged face's mouth with

a preset value of a Blendshape object for each of these shapes. The loop starts by setting the value of the Blendshape object to a blended value between the shape of the rigged face's mouth (`riggedFace.mouth.shape[shape]`) and the current value of the Blendshape object (`Blendshape.getValue(PresetName[shape])`). The blend is done using the lerp function, which stands for linear interpolation. This function takes three arguments: the start value, the end value, and the weight of blending. In this case, the weight is set to 0.5, resulting in an even blend between the two values. Once the blended value is calculated, it is set to the Blendshape object using the PresetName as the key. This process is repeated for each shape in the array, allowing the developer to easily blend the shape of the rigged face's mouth with preset values for multiple shapes.

3.3.4 Blinking eyes

Listing 2: Blinking of the eyes

```
1    riggedFace.eye.l = lerp(clamp(1 - riggedFace.eye.l, 0, 1),
         Blendshape.getValue(PresetName.Blink), 0.5)
2    riggedFace.eye.r = lerp(clamp(1 - riggedFace.eye.r, 0, 1),
         Blendshape.getValue(PresetName.Blink), 0.5)
3    riggedFace.eye = Kalidokit.Face.stabilizeBlink(riggedFace.eye,
         riggedFace.head.y)
4    Blendshape.setValue(PresetName.Blink, riggedFace.eye.l)
```

This pseudocode 2 is a code block that performs blinking animation on a VRM model. The code uses the Lerp function, the clamp function, and the stabilizeBlink function to animate the eyes of the rigged face.

The first two lines of the code use the Lerp function to blend the current value of the left and right eye shapes (`riggedFace.eye.l and riggedFace.eye.r`) with a preset Blink value from the Blendshape object. The Lerp function takes the clamp of the inversed eye shape values (`1 - riggedFace.eye.l and 1 - riggedFace.eye.r`), which clamps the values to between 0 and 1, and the preset Blink value, which is obtained using `Blendshape.getValue(PresetName.Blink)`. The weight of the blend is set to 0.5, resulting in an even blend between the two values.

The third line of the code stabilizes the blinking animation of the eyes by calling the stabilizeBlink function from the `Kalidokit.Face` module. This function takes the current eye shape (`riggedFace.eye`) and the head's y position (`riggedFace.head.y`)

as arguments and returns a stabilized eye shape.

Finally, the fourth line of the code sets the Blink value of the Blendshape object to the stabilized left eye shape (`riggedFace.eye.l`). This will result in a smooth and stable blinking of the eyes on the VRM model.

In summary, this pseudocode is a code block that animates the eyes of a VRM model by blending the current eye shape with a preset Blink value, stabilizing the blinking animation, and updating the Blink value of the Blendshape object. The code uses the Lerp function, the clamp function, and the stabilizeBlink function to create a smooth and stable blinking animation on the VRM model.

3.3.5 Positioning the model

Listing 3: Position of the model

```
1       function rigPosition(name, position={x:0, y:0, z:0}, dampener=1,
            lerpAmount=0.3) {
2           if (currentVrm is not defined) {
3             return;
4           }
5           Part = currentVrm.humanoid.getBoneNode(name from
                THREE.VRMSchema.HumanoidBoneName);
6           if (Part is not defined) {
7             return;
8           }
9           vector = new THREE.Vector3(position.x * dampener, position.y * dampener,
                position.z * dampener);
10          Part.position.lerp(vector, lerpAmount);
11      }
12
13        oldLookTarget = new THREE.Euler();
14        function rigFace(riggedFace) {
15          if (currentVrm is not defined) {
16            return;
17          }
18          rigRotation("Neck", riggedFace.head, 0.7);
19
20          Blendshape = currentVrm.blendShapeProxy;
21          PresetName = THREE.VRMSchema.BlendShapePresetName;
22      }
```

This pseudocode 3 is a code block that positions a VRM model according to a set of defined parameters. The code block consists of two functions: rigPosition and rigFace.

The **rigPosition function** takes three arguments: name, position, dampener, and lerpAmount. The name argument is the name of the part of the VRM model that needs to be positioned, the position argument is an object with x, y, and z properties that represent the desired position of the part, the dampener argument scales the

position values, and the lerpAmount argument is the weight of the blend between the current position and the desired position. The function starts by checking if the currentVrm is defined, and if it's not, the function returns without executing further code. If currentVrm is defined, the code uses the `humanoid.getBoneNode` method to retrieve the part specified by the name argument. If the part is not defined, the function returns without executing further code. If the part is defined, the code creates a new `THREE.Vector3` object with the desired position values (scaled by the dampener argument), and the Part.position property is blended towards the desired position using the lerp method. The weight of the blend is set by the lerpAmount argument.

The **rigFace function** takes one argument: riggedFace, which represents the face of the VRM model. The function starts by checking if the *currentVrm* is defined and if it's not, the function returns without executing further code. If *currentVrm* is defined, the function calls the rigRotation method and passes two arguments: *Neck* and *riggedFace.head*, with a third optional argument of 0.7. This rotates the neck of the VRM model according to the head position. The function also defines the Blendshape variable and assigns it to the `blendShapeProxy` property of the *currentVrm* object. The `PresetName` variable is set to the `BlendShapePresetName` property from the `THREE.VRMSchema` object.

3.3.6 Animating the body

Listing 4: Animating the rest of the body

```
    if (pose2DLandmarks && pose3DLandmarks) {
    riggedPose = solvePose(pose3DLandmarks, pose2DLandmarks, "mediapipe",
        videoElement);
    rotate("Hips", riggedPose.Hips.rotation, 0.7);
    position("Hips", -riggedPose.Hips.position.x, riggedPose.Hips.position.y + 1,
        -riggedPose.Hips.position.z, 1, 0.07);

    rotate("Chest", riggedPose.Spine, 0.25, 0.3);
    rotate("Spine", riggedPose.Spine, 0.45, 0.3);

    rotate("RightUpperArm", riggedPose.RightUpperArm, 1, 0.3);
    rotate("RightLowerArm", riggedPose.RightLowerArm, 1, 0.3);
    rotate("LeftUpperArm", riggedPose.LeftUpperArm, 1, 0.3);
    rotate("LeftLowerArm", riggedPose.LeftLowerArm, 1, 0.3);

    rotate("LeftUpperLeg", riggedPose.LeftUpperLeg, 1, 0.3);
    rotate("LeftLowerLeg", riggedPose.LeftLowerLeg, 1, 0.3);
    rotate("RightUpperLeg", riggedPose.RightUpperLeg, 1, 0.3);
    rotate("RightLowerLeg", riggedPose.RightLowerLeg, 1, 0.3);
}
```

The pseudocode ,4 you have posted describes an animation process for a VRM body. The code checks if the *pose2DLandmarks* and *pose3DLandmarks* are available. If they are both available, the code calculates the rigged pose using the function `solvePose` with the *pose3DLandmarks*, *pose2DLandmarks*, *mediapipe*, and *videoElement* as input parameters. The function `solvePose` returns the rigged pose, which is then stored in the variable `riggedPose`. The code then rotates and positions various body parts of the VRM model based on the values obtained from the rigged pose. For example, the hips are rotated based on the value of `riggedPose.Hips.rotation` with a weight of 0.7, and their position is changed to `x: -riggedPose.Hips.position.x, y: riggedPose.Hips.position.y + 1, z: -riggedPose.Hips.position.z` with a weight of 1 and a smoothing factor of 0.07. Similar operations are performed for the chest, spine, arms, and legs. The rotation and position of each body part is performed using two separate functions: `rotate` and `position`. The first parameter of each function specifies the name of the body part to be animated, while the remaining parameters specify the values used for rotation or position, weight, and smoothing factor. In this way, the VRM model's body is animated according to the solved pose information.

3.3.7 Animation of both hands

Listing 5: Animating the left hand

```
    if (leftHandLandmarks) {
        # Obtain the rotation information of the left hand
        riggedLeftHand = Hand.solve(leftHandLandmarks, "Left");

        # Apply the rotation to the left hand rig
        rigRotation("LeftHand", {
          z: riggedLeftHand.LeftWrist.z,
          y: riggedLeftHand.LeftWrist.y,
          x: riggedLeftHand.LeftWrist.x,
        });

        # Apply the rotation to each of the finger bones
        rigRotation("LeftRingProximal", riggedLeftHand.LeftRingProximal);
        rigRotation("LeftRingIntermediate", riggedLeftHand.LeftRingIntermediate);
        rigRotation("LeftRingDistal", riggedLeftHand.LeftRingDistal);
        rigRotation("LeftIndexProximal", riggedLeftHand.LeftIndexProximal);
        rigRotation("LeftIndexIntermediate", riggedLeftHand.LeftIndexIntermediate);
        rigRotation("LeftIndexDistal", riggedLeftHand.LeftIndexDistal);
      }
```

The provided pseudocode 5 animates the left (same for the right) hand of a VRM character. It starts by checking if the left hand landmarks have been detected. If they have, the code then calculates the rotations of the left hand by using the

`Hand.solve` function, which takes in the left hand landmarks and the string `Left`. The resulting rotations are then applied to the left hand rig and each of the finger bones using the `rigRotation` function. For the left hand rig, the function is called with the string `LeftHand` and the rotation values for the x, y, and z axes, obtained from the `riggedLeftHand` object. Similarly, the rotations of each finger bone are applied by calling the `rigRotation` function with the names of the bones, such as `LeftRingProximal`,`LeftRingIntermediate`, and `LeftRingDistal`, and the corresponding rotation values obtained from the `riggedLeftHand` object. The same process is repeated for each of the other finger bones.

4 UI-Design

5 Umfeldanalyse

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. Citing [10] properly.

Was ist eine GUID? Eine GUID kollidiert nicht gerne.

Kabellose Technologien sind in abgelegenen Gebieten wichtig [11].

6 Umsetzung

Siehe tolle Daten in Tab. 1.

Siehe und staune in Abb. 7. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

	Regular Customers	**Random Customers**
Age	20-40	>60
Education	university	high school

Table 1: Ein paar tabellarische Daten

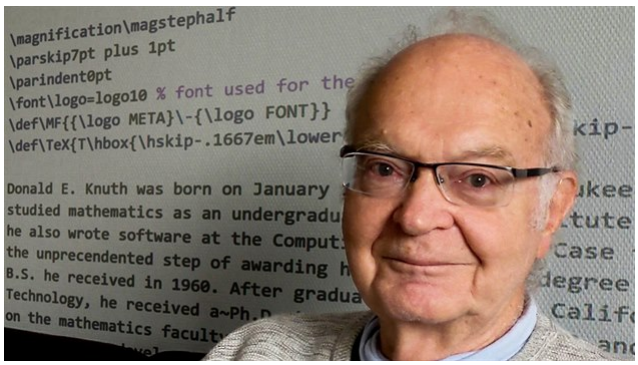

Figure 7: Don Knuth – CS Allfather

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus. Dann betrachte den Code in Listing 6.

Listing 6: Some code

```
# Program to find the sum of all numbers stored in a list (the not-Pythonic-way)

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

7 Zusammenfassung

Aufzählungen:

- Itemize Level 1
  - Itemize Level 2
    - Itemize Level 3 (vermeiden)

1. Enumerate Level 1
   a. Enumerate Level 2
      i. Enumerate Level 3 (vermeiden)

Desc Level 1

Desc Level 2 (vermeiden)

Desc Level 3 (vermeiden)

Glossar

GUID Globally Unique Identifier

Bibliography

[1] Y. Song, „A Review on Different Kinds of Artificial Intelligence Solutions in TCM Syndrome Differentiation Application," *Hindawi Publishing Corporation*, Vol. 2021, Nr. 6654545, 2021.

[2] IBM, „IBM Research - Zurich," IBM Website. Online verfügbar: http://www.research.ibm.com/zurich/

[3] A. G. Eberhard Schöneburg, Nikolaus Hansen, *Neuronale Netzwerke : Einführung, Überblick und Anwendungsmöglichkeiten*, 3. Aufl. Markt und Technik, 1990.

[4] H. B. Xiaoyi Jiang, *Dreidimensionales Computersehen Gewinnung und Analyse von Tiefenbildern.* Springer, 1997.

[5] G. L. MediaPipe Team, „MediaPipe," MediaPipe Website/Library. Online verfügbar: https://mediapipe.dev/

[6] Y. N. W. Bo Pang, Erik Nijkamp, „Deep Learning With TensorFlow: A Review," Tensorflow Article. Online verfügbar: https://journals.sagepub.com/doi/abs/10.3102/1076998619872761

[7] S. Ochella und M. Shafiee, „Performance Metrics for Artificial Intelligence Algorithms Adopted in Prognostics and Health Management of Mechanical Systems," *IOP Publishing*, Vol. 2021, Nr. 012005, 2021.

[8] three.js, „GLTFLoader," three.js Library/Website. Online verfügbar: https://threejs.org/docs/#examples/en/loaders/GLTFLoader

[9] ——, „three.js," three.js Library/Website. Online verfügbar: https://threejs.org/r

[10] P. Rechenberg, G. Pomberger *et al.*, *Informatik Handbuch*, 4. Aufl. München – Wien: Hanser Verlag, 2006.

[11] Association for Progressive Communications, „Wireless technology is irreplaceable for providing access in remote and scarcely populated regions," 2006, letzter Zugriff am 23.05.2021. Online verfügbar: http://www.apc.org/en/news/strategic/world/wireless-technology-irreplaceable-providing-access

List of Figures

List of Tables

Quellcodeverzeichnis

Anhang