E(x|g)o: Comparing Visual Perspectives on Guidance Visualisations for Motor Learning in Virtual Reality

Stefan Paul Feyer
HCI Group, University of Konstanz

Maximilian Dürr
Supervisor

Master's Project Report
March 2020

Contents

Abstract

abstract

1 Introduction

1.1 Introduction

2 Technology Evaluation

The hardware setting consists out of three main parts. First an Virtual Reality Head Mounted Display which serves as the window to the virtual world for the participant. Secondly, a motion capturing system, which serves as translation of the real world movements in the virtual world. Thirdly, the assets with which the participant of the study will interact with. This chapter discusses the requirements these three aspects have to fulfil to have a reliable and suitable study design.
In the early stages of planning a study, it is very important to choose the hardware wisely. The hardware is the base on which the software is build up and both needs to fulfil requirements to be usable in a study. For example the accuracy must be high enough to measure what want to be measured and also it must be as reliable as possible, to not disrupt the study procedure. For this reason hard and software were analysed. The hardware components are the HMD, Motion Tracking devices and the assets the participant will interact with and for the software, an engine and its plugins.

2.1 Hardware

HMD

In 2013, the first next generation HMD was on market for sale. The Oculus Rift DK1 had given the first glimpse on what will be possible with this new Virtual Reality headsets. The next big step was the evolution of graphics cards and with the GTX 1080 released in May 2016, enough graphics power was widely available to run state of the art head sets. Since then, more and more headsets of multiple companies made it to market maturity and today we are spoilt of choice. Meanwhile there are many headsets that fulfil the requirements to teach motor learning in virtual reality. They are fast, exact, have plenty of pixels with reasonable refresh rates and a wide field of view. But they differ in the tracking technology used to identify they position and orientation in space, as well as other aspects like the information transfer to an pc or the possibility of wearing glasses beneath it. The possibility to wear glasses beneath the head mounted display is important, because

no one is excluded to participate in the study for the reason wearing glasses. For this project the cable does not play a big role. Since all headsets worth considering fulfil the requirements to conduct a study on motor learning, the decision was mostly made from the possibility to wear glasses and having the same coordinate system then the motion tracking technology, which guarantees easier and a more stable environment. The Valve Index headset currently has with the best performance and fulfils the additional requirements. Though the choice here was easy[1]. A small comparison of VR HMDs can be found in Table

Motion Tracking

In contrast to the VR HMD, the choice for motion tracking hardware more difficult.

The requirements for the motion capturing system are as follows:

- **Accuracy**: to compare movements, the accuracy should be under 1cm
- **Large movement area**: motor learning includes movement in space. The tracked area should be at least 20qm
- **Capable for humanoid movements**: tracking of objects and movement differ fundamentally. For motor learning, humanoid movements are necessary.
- **Freedom of movements**: during motor learning, a human can move in all directions and all possible postures conceivable. The motion tracking system must cover this.
- **Sufficient tracking points**: for measuring movements, enough tracking points must be provided for comparing the movements.
- **Extendable**: assets must be able to be tracked as well.
- **Reliable, study safe setup**: jitter or calibration loss can void a study.
- **No coordinate system matching**: matching of different coordinate system for e.g. HMD and the human body is an error source which should be excluded.

[1]Meanwhile the Pimax 8K is on sale. From a technical view this is a headset outperforming the Valve Index. A test of this hardware is planned and if it proves to be suitable the study will be conducted with the Pimax 8K

There are two main classes of motion tracking systems: inside-out and outside in. Both of them have pros and cons. Inside-out tracking uses active sensors on the body to track. Perception Neuron [2] uses the magnetic field of the earth to identify position and orientation changes. But magnetic sensors are prone to metal. Other systems [3] use accelerometer or gyroscopes or a combination to track the movements. But there is still the drift problem: a position is determined by the previous position and the error adds on the error before. The longer the capturing, the greater the error. On the plus side, the tracked area is potentially infinite and the sensors are rather small. On the other side, outside-in tracking system use external tracking devices to track points on the body in question. These systems deliver an absolute position and orientation in the tracking space and do not have the drift issue. If accuracy is the measurement in question, inside-out tracking systems are always second to outside-in tracking systems. For this reason, in the following only outside-in tracking systems are discussed.

Kinect

Microsoft Kinect 2.0 is the successor of the 1.0 version released in 2014. It uses a RGB camera and an IR camera to calculate a skeleton of a human body in front of it. It is very easy to setup and use. Also the tracking area is large enough. But the participant needs to face the Kinect at all time to have a reliable, jitter free tracking. An evaluation of the Kinect 2.0 revealed that it is not suitable for motor learning. Compare test video:

OptiTrack

OptiTrack is an optical motion tracking system. Multiple infrared cameras track reflective markers on the human body, calculating a skeleton.

Vive Tracker

Assets

baseplates, box, table

2.2 Software

[2] yxc
[3] yxc

3 Implementation

3.1 Hardware

The first step regarding the implementation is to prepare the hardware. The main elements are:

- 4x Lighthouse 2
- VR HMD: Valve Index
- 7x Vive Tracker 2
- 7x Vive Tracker mounts
- PC
- OptiTrack suit
- Table
- Box

Lighthouse

The so called *Lighthouse 2* by Valve, also called *Base Stations* are rectangular boxes in size of 7,5x7,5x6 cm. They emit a fast moving laser beam. With two base stations, the HMD, Vive Trackers and Controller can determine the exact position and orientation in 3D space in real time. The Lighthouse enables E(x|g)o to track all actors. The setting includes 4 Base Stations in each corner of the the tracked volume. Two Base Stations would be enough to track the actor in the volume, but since the system is optical it is prone to occlusion. To overcome this issue, 4 Base Stations are used. Compare yxc.

VR HMD: Valve Index

The Valve Index VR HMD let the user dive into the Virtual World. The user wears this HMD on the head. The HMD utilises the Lighthouse to determine its position and orientation in the tracking volume. The Valve Index is cable bound and transmits its data (for example internal gyroscope data) via USB to the PC. The PC transmits the visual information to the HMD via Display Port. Finally, the HMD needs a power supply. All connection is transferred by a single cable that spreads at the end. Compare yxc

Vive Tracker 2

Vive Tracker 2 are star shaped elements, that utilise the Lighthouse to determine its position and orientation in the tracking volume. The Vive Tracker 2 transmits its data via Bluetooth. For each Vive Tracker a special dongle mus be plugged in the PC on a USB port.

Vive Tracker Mounts

The Vive Tracker must be attachable to objects the the user. For this purpose mounts in two different sizes were manufactured by a 3D printer. The mounts are rather high in size which leads to wobbly behaviour. For the study a flatter design is more advisable. Compare yxc old mount and new mount. The mounts are fixated by a M6 screw to the Vive Tracker. On the backside of the mounts Velcro is attached.

PC

The PC must be capable of handling the VR HMD. The main bottleneck is the graphics card. The used PCs graphic card is a GTX 2080. The PC must be connected to the HMD. Furhtermore, the PC must provide enough USB Ports for HMD, periphery devices, one per Vive Tracker.

OptiTrack suit

During the development of such a system, a highly flexible setup is advisable to adjust the hardware easily. The surface OptiTrack suit is completely covered by Velcro. The counterpart is attached to the Vive Tracker which allows to rearrange the Vive Tracker fast and easy. For the final study setting, special fixation system could be provided for more comfort. Furthermore, the Vive Trackers need to be alinged always the same way. The suit helps with this issue, too.

Box & Table

For developing a cardboard box was used. On the box, a Vive Tracker is fixated. Same goes with the table. With these Vive Trackers, the box and table can be made visible in the software, allowing the user to interact with.

3.2 Software

SteamVR Principles

SteamVR is a driver handling the connected Tracker, HMD and Controller. After a calibration, up to 16 devices can be added to one instance. In this case these devices are: 4x Lighthouse, 1x Valve Index HMD, 7x Vive Tracker and 1x controller. The driver provides the position and orientation to programs utilising this information. In this case, the Unity asset SteamVR access this information.

FinalIK Principles

FinalIK is a Unity package providing necessary calculations to represent a human body based on the position of view points of a real body. yxc todo

VRIK

VRIK is a part of FinalIK with the focus of rendering human body's for virtual reality. VRIK utilises 5 Vive Tracker: 2 on the feet, 1 one the hip and 2 on the hands. The HMD is the last reference point. Based on these 6 point, it renders solves the limbs of a human body and renders a humanoid character.

Vive Tracker Matching

In Unity, a Vive Trackers position and orientation can be assinged to a GameObject with the script*SteamVRTrackedObject* compare 3.1 middle. In this script a device ID is used to be associated with a physical Vive Tracker. Unfortunately, every time SteamVR driver is started, the device IDs change. For example device ID 1 was associated with the Vive Tracker placed on the left feet of the user. On the next startup, device ID 1 can be associated with lighthouse bas station. This leads to no study safe setup and must be corrected. To solve this issue, first all device hardware IDs of the used devices must be read out. Secondly, the hardware id must be matched to specific devices. Then, at the startup of the system, an algorithm reads out all hardware ids and assign it to the correct GameObject by associating the correct devid ID.

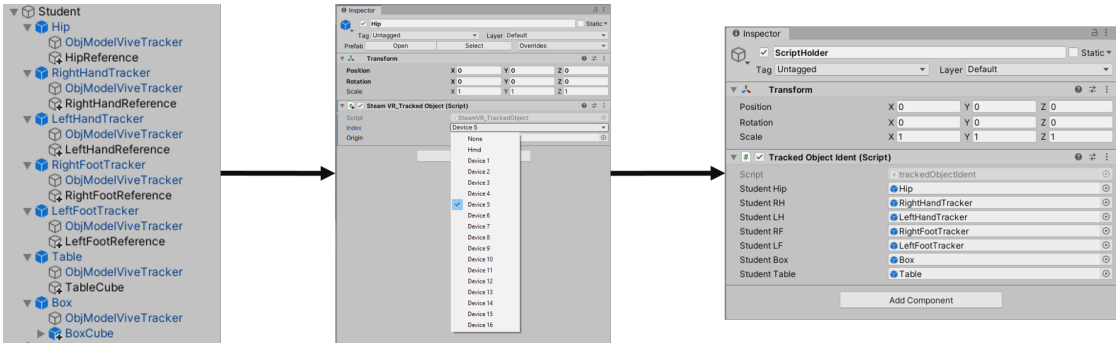

Figure 3.1:

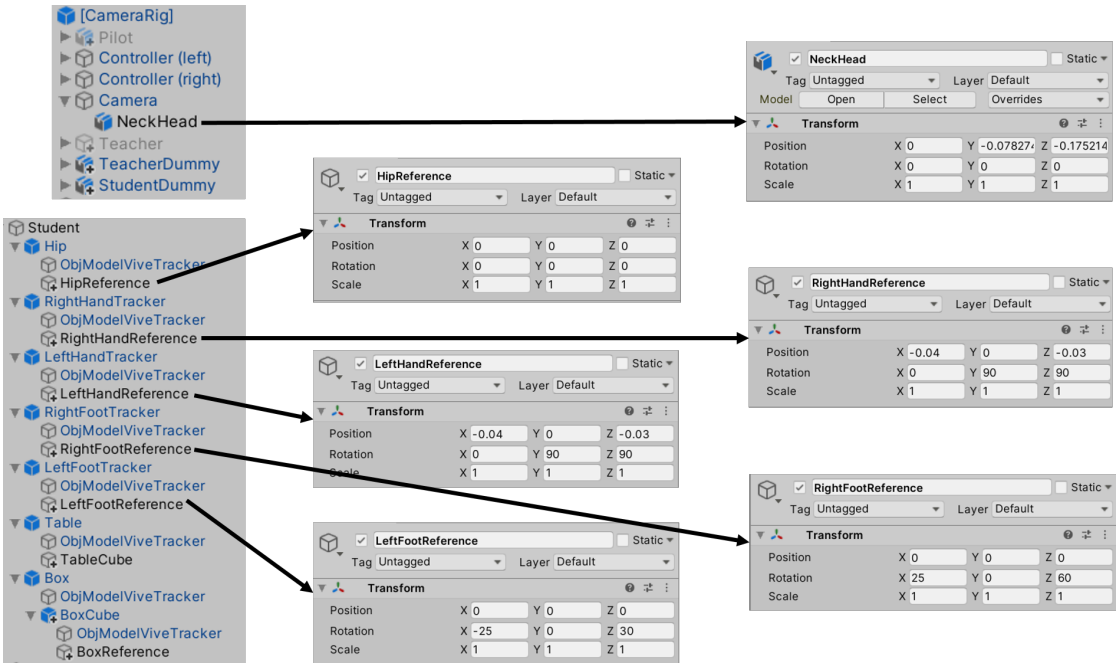

Figure 3.2:

VRIK Calibration

VRIK solves the limbs based on the above mentioned tracking points. The head reference is a child of the *camera* provided by the SteamVR asset. It holds position and rotation and is the most important input for the solver. The VRIK solver takes this position as absolute and solves everything else based on this position. The position but not rotation overrides even overrides parent transforms. This fact influences the further handling in Unity massively: to transform a VRIK, the GameObject can not be translated to a specific position. This can only happen

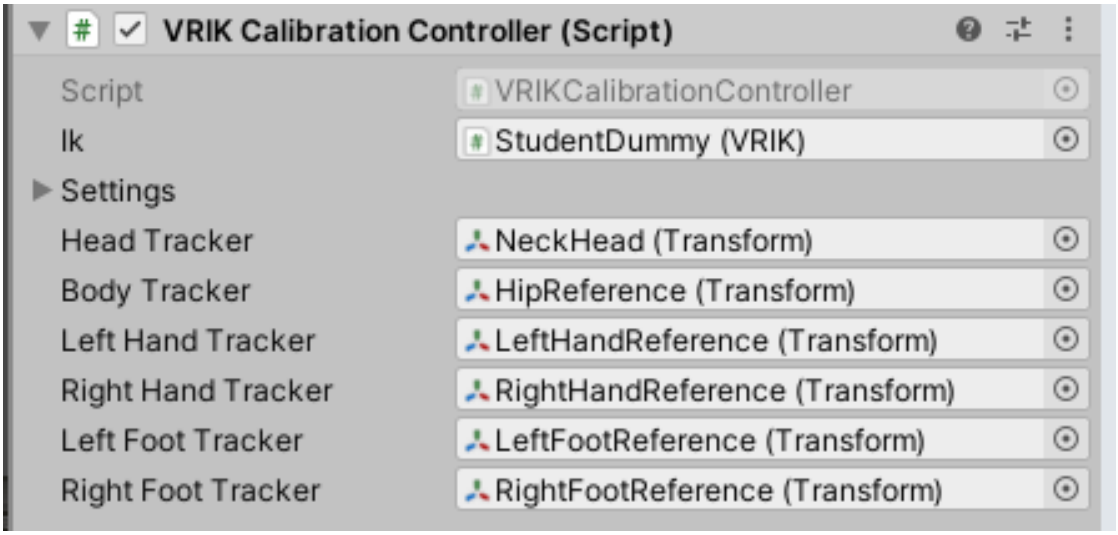

Figure 3.3:

by transforming the references.
Left and right hand position needs to be adjusted, because their root lies inside the body in the middle of the joint. To transform the References accordingly, they are a child to the tracker reference. From this root, the references are shifted 4 cm towards X and 3 cm towards Y. Additionally, the rotation needs to be adjusted by 90° in Y and Z. With this, the tracking point is located in the middle on the back of the hand.
Left and right feet need adjustments to in terms of position and rotation. The Vive Tracker is located at the top of the feet, which is fine with the root of the VRIK reference. But the slope of the trackers and forward direction must be adjusted. The left reference needs a tilt of -25°, the right reference by +25°. For the correct direction, the left reference needs a rotation by 30° on Z-axis, the right by 60°.
The hip does not need further shifts.
With the correct position of the references, the references can now be assigned to an VRIK instance. This works with an VRIK calibration controller. The calibration controller takes the references and assigns it to the VRIK instance. In the same step the humanoid character is sized to the references.

Student Rendering

The rendering of the students humanoid character is based on the live tracking data of the Vive Trackers. As seen above, the Trackers get associated to a GameObject e.g. Hip. Relative to this GameObject, a reference is created. this reference is

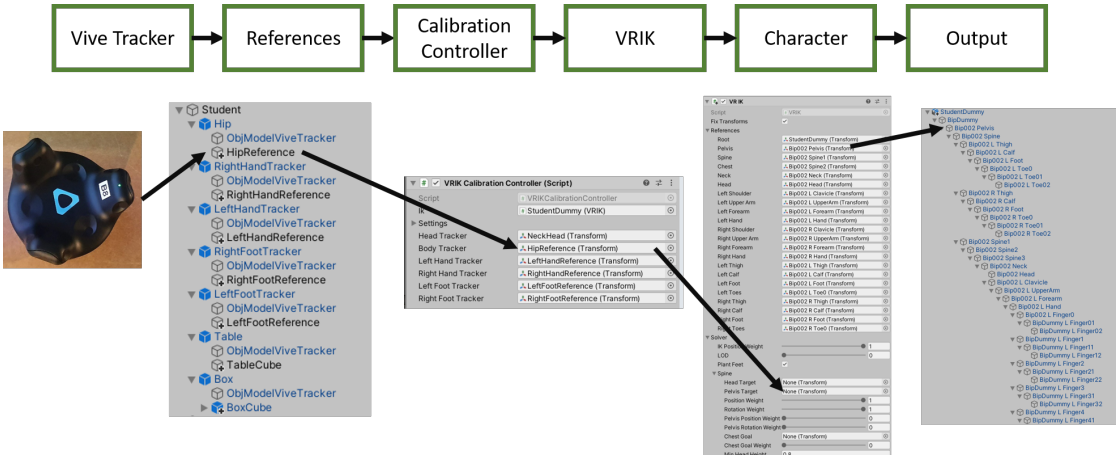

Figure 3.4:

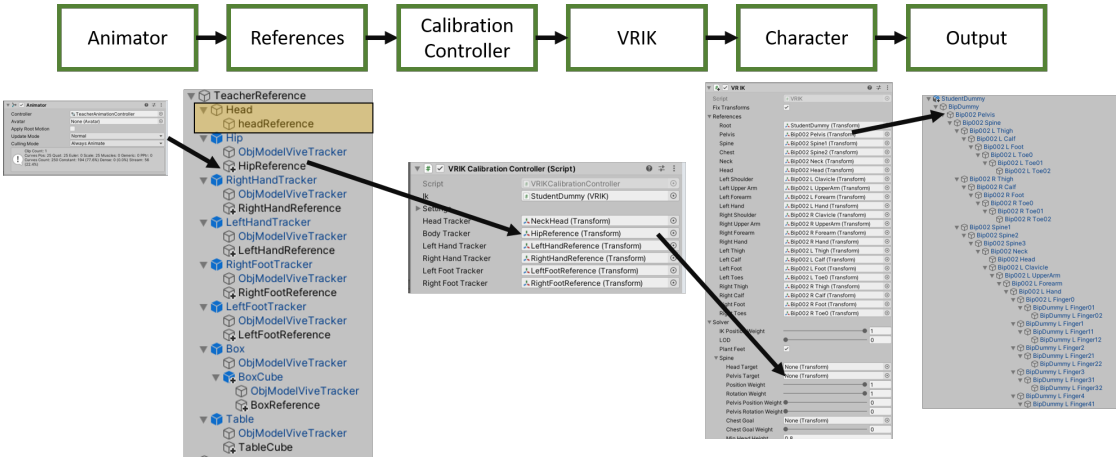

Figure 3.5:

passed to the calibration controller. The calibration controller sets the corresponding references in the VRIK solver. So far nothing new.
Now the VRIK takes an humanoid character and binds it muscles. then the solving begins and the algorithm animates the associated humanoid character. This character itsself consists out of a hierarchy of bones and muscles. These bones and muscles are overlaid by a mesh renderer which produces the desired output.

Teacher Rendering

The naive approach of rendering the teacher is to record a teacher and then just play the animation. But recording a humanoid animation is not natively supported

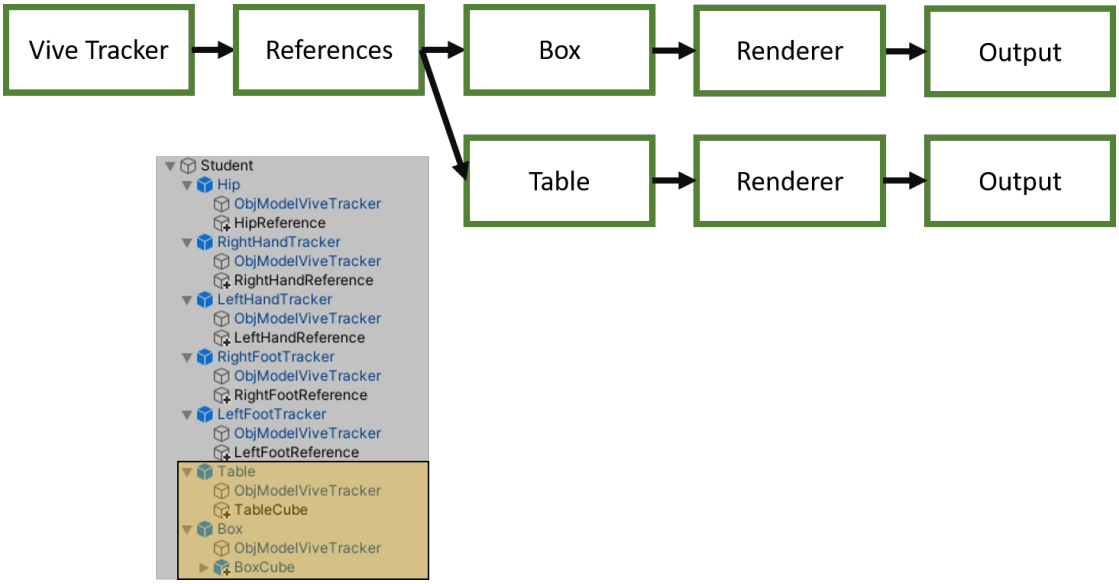

Figure 3.6: asset rendering pipeline

by Unity and would require an third party tool to do so. This leads to unwanted differences in the representation of the teacher. To ensure, that teacher and student produce comparable renderings, the same technique has to be utilised. To accomplish this requirement, the movements are recorded by recording the students tracker position and then pass this as animations to a similar shaped hierarchy of GameObjects, that in turn are used are passed the an VRIK solver. This is why the rendering pipeline of the teacher differs from the students rendering pipeline. The process for the teacher is as follows: For the teacher, a reference hierarchy needs to be created. The calibration above applies, but the head must be extracted from the camera and transformed into an own OameObject with the heads reference turn by 90° on the Y and Z axis. On the parent *TeacherReference* an *AnimatorController* is attached. This AnimatorController plays the pre recorded animation and transforms the references of the children accordingly. Now these children can be used like in the students rendering pipeline and passed to the CalibrationController. Afterwards the rendering pipeline is as seen in the students pipeline.

Assets Rendering

Finally, the objects the student and the teacher are interacting, namely the table and the box, need to be rendered. Rendering of these object is less complex. The students objects get attached by a Vive Tracker. The position and rotation of the

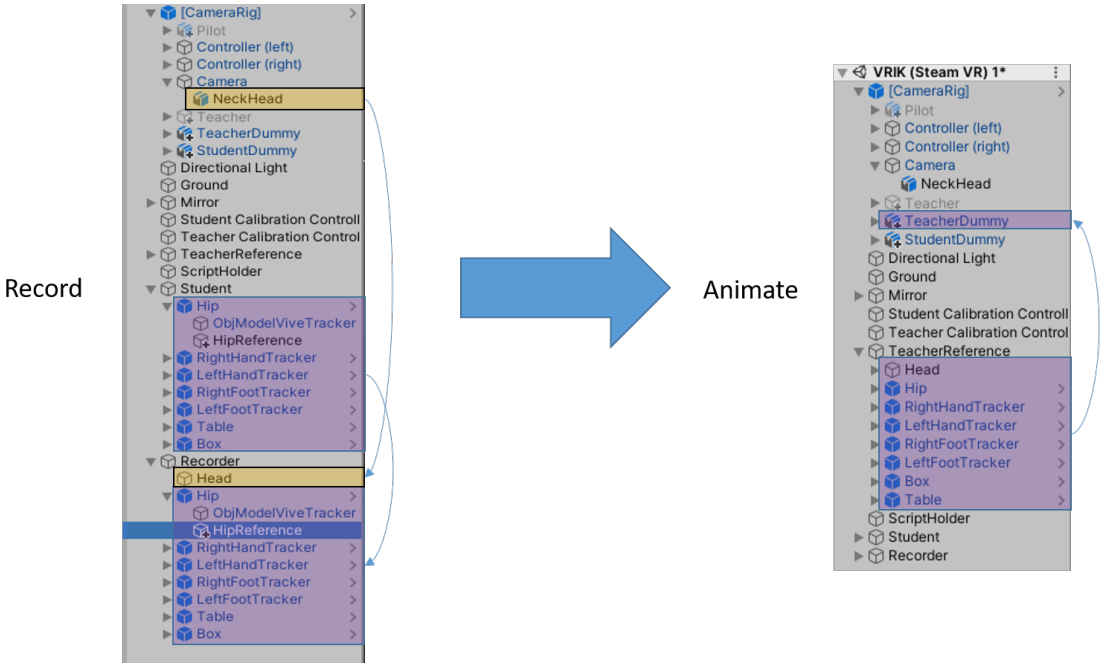

Figure 3.7: asset rendering pipeline

Tracker is bound to a cube, which is scaled to the size of the box or the table. The teachers assets are also cubes scaled to the size of the cube and the table. But the position and rotation are set by the animator controller.

Recording of Movements

As mentioned above, recording of humanoid movements are not trivial. The solution is to create a GameObject with a similar hierarchy. This GameObject is called *Recorder*. It is similar structured like the student. The references are holding a script which sets the transforms them to the transforms of the student. Additionally, the head reference, the box reference and the table reference. The transforms of these references are also copied from the real elements. This is necessary to have one single animation that controls the teacher. Otherwise time shifts between animations could be possible. Now the complete *Recorder* hierarchy can be recorded by the *Unity Recorder*. The outcome is an animation that produce the references necessary to render the teacher like described above.

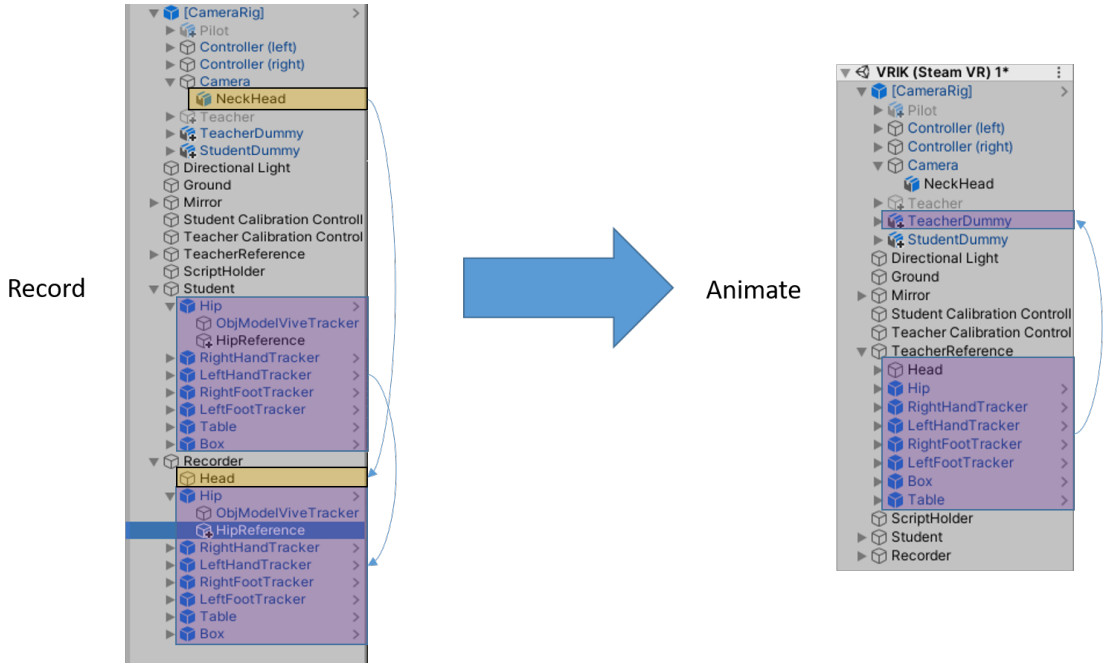

Figure 3.8: asset rendering pipeline

Resize

The teacher and the student are most likely to have different body sizes. especially in the ego-centric perspective, the size of the teacher must match the size of the student. But the naive approach of resizing the teachers humanoid character to the size of the student leads again not to a satisfying result. This is because the head position is absolute and not changeable for the VRIK solver. Resizing the teacher leads to floating or swaged representations. The solution for this is to resize the parent of the the teachers references, and then pass the already scaled references to the calibration controller. But this leads to another issue: the size of the student can only be assessed after the calibration controller did his work, assinged the references to the VRIK solver and sized the representation. To overcome this issue a 2-step body size calibration is necessary. First, the the students calibration controller performs the calibration of the student. Then the students hight is measured and compared to the teachers size and stored in δy. δy describes the percentage of size difference between teacher and student. Afterwards, the teachers references root is scaled by δy and gets reassigned to the calibration controller, which itself performs the task again. With this, the teacher has the same size as the student.

3.3 Visual Perspectives

Ego-Centric

The nature of an ego-centric perspective raises one big issue to tackle: the student has "to be in the teacher" in any point of time. If the teacher now wants to indicate a movement, meaning a translation of the own position in space, it is indicating this movement by moving away from the student. This leads inevitable to an non ego-centric perspective. To solve this issue, a closer look on the definition of the visual perspectives can help: the visual perspectives are represented by a continuum. On one extreme the ego-centric visual perspective is located, while on the other extreme the exo-centric perspective is located. With moving from one extreme to the other, the tethering distance[1] is changing. A tethering distance of 0 indicates a pure ego-centric perspective while a greater tethering distance means shifting towards the exo-centric extreme. Following the nature of an continuum, a slightly larger tethering distance than 0 is still an ego-centric perspective. The task is now to choose a value for the tethering distance that still it still is an ego-centric perspective, but at the same time let the student be able to interpret the teachers indication of movement. Based on empirical values the tethering distance is set to 30cm. It still feels ego-centric, but the indication of the movement of the teacher is clear to recognise. But a hard threshold would lead to non fluid demonstrations of the teacher, which is hard to follow for the student. For this reason, the tethering is set to a frame of 15cm to 30cm with a variable, interpolated speed of the teachers demonstration. This means, between 0cm and 15cm the teachers movement demonstrations are shown in normal speed. Between 15cm and 30cm the speed linearly decreases and stops completely at a tethering distance of 30cm. This makes the teachers movements easily to follow by the student with no interruptions.
In the following, the implementation for this mechanic is described. First the teacher must be translated to the position of the student. Again, the naive approach of just shifting the teachers avatar on the students position doesn't fulfil the requirements, because the VRIK solver takes the position of the references as absolute. Though, the teachers references parents zero holding the animator has to be transformed. For this, the distance between the parent of the references zero of teacher an student is calculated and then the teachers references parent is shifted by this distance. Secondly, the animation speed is set by the by the distance between the student and the teacher. For this calculation, the hip references of both are taken into consideration. Is the difference of these two points below 0.15 the animation speed of the teacher is 1. If the distance is greater than 0.3 the

[1]The tethering distance is the distance between the eyes anchor point and the camera observing the character in question.

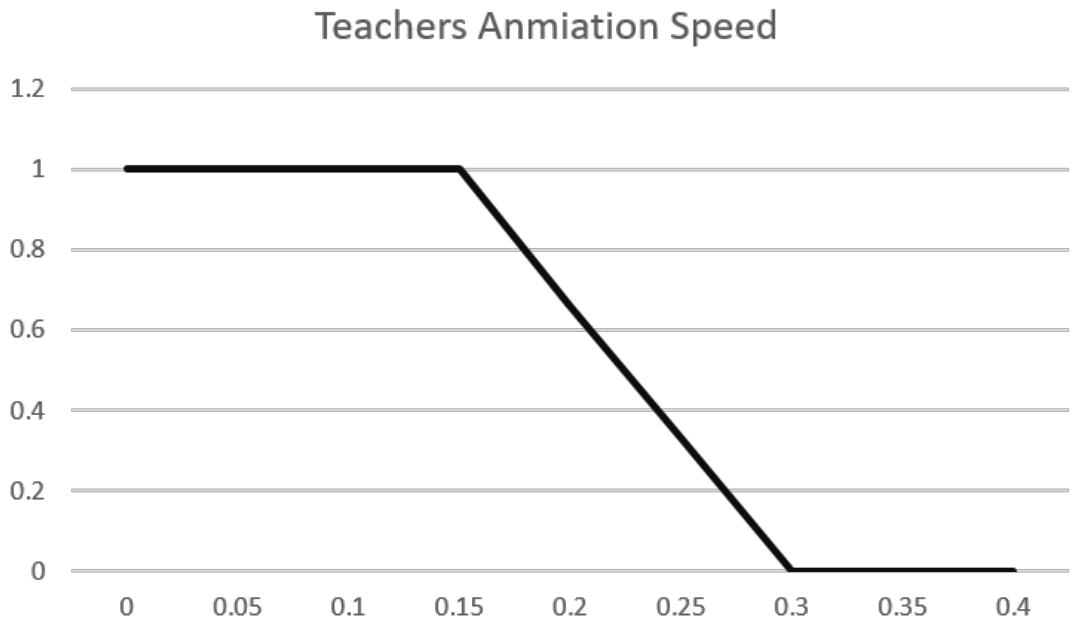

Figure 3.9: Teachers animation speed, linear interpolated. yxc add axis names

animation speed is 0. Between 0.15 and 0.3 the animation is a linear interpolation. The code show this:

Exo-Centric

The exo-centric visual perspective is less complex than the ego-centric visual perspective. The teacher is located outside of the student and the student can choose the own position to observe the teacher. But in this condition the student can turn away from the student by following the instructions of the teacher. To visualise this issue, follow this scenario: teacher and student standing side by side and looking into the same direction. The student stands left of the teacher. If the teacher turns left and the student also turns left, the student can no longer see the teacher and follow the instructions. But even when the teacher is turning to the right, the student sees the teacher from the back, unable to see what the teacher is doing in front of the teachers body. One could argue that in real world scenarios the same issue exists. But in real world scenarios where this issue is likely, it is also likely that in the room are mirrors, allowing to see areas that are coved by the teachers body. With this argumentation multiple representations of the teacher are introduced. Instead of having only one instance of the teacher, four instances on different positions around the student are present. By empirical values, the

teacher is translated by Now the the student has multiple angles on the teacher, which overcome the mirror issue[2], too.

Ego-Centric & Exo-Centric

The combination of the ego-centric and exo-centric perspective leads to the third visual perspective. Like the name indicates, the teacher can be seen from the ego-centric perspective and the exo-centric perspective simultaneously: teacher stands outside and inside of the students body. To achieve this, the two above visual perspectives are combined. Starting from the above described exo-centric visual perspective with its four teachers, a fifth teacher is introduced and placed inside of the student with the same mechanics as in the above described ego-centric perspective. Eventually, the animation speed of the ego-centric teacher must be applied to the exo-centric teachers. This happens by the script *putTeacherIntoStudent.cs* which holds references to all animation controllers. The script synchronises all animations and changes the speed of the animation playback.

Augmented Exo-Centric

In the augmented exo-centric perspective, the student stands inside of the the teacher. While the number of teachers remains at 4, the student now needs to have multiple representations. One at the real world location, representing the ego-centric perspective, and four more in every teacher. The formation of the exo-centric perspective remains. Because no ego-centric perspective is present, the animation are not needed to have speed interpolation.

[2]Mirror issue: a teacher in standing in front of the student forces the student to move the e.g. left arm when the teacher moves the right arm. Compare seminar thesis for more details

4 Final System

4.1 fs

5 Conclusion

5.1 concl

Bibliography

List of Figures

List of Tables