POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Modular Sign Language synthesis using high fidelity 3D avatars

Supervisors
Prof. Andrea Bottino
Roberto Iacoviello
Alberto Ciprian
Davide Zappia

Candidate
Claudio Giuseppe Messina

October 2025

Acknowledgements

Vorrei ringraziare tutte le persone che mi sono state accanto durante il mio percorso universitario. Se sono arrivato a questo punto è indubbiamente anche grazie al loro immenso supporto.

Innanzitutto ringrazio il Prof. Andrea Bottino, Roberto Iacoviello, Alberto Ciprian e Davide Zappia per avermi guidato nella realizzazione di questa tesi e per la loro disponibilità nel condividere con me le loro conoscenze.

Il grazie più grande va sicuramente ai miei genitori, mia sorella e tutta la mia famiglia, che mi hanno sostenuto e incoraggiato fin dal primo giorno, spingendomi a superare i miei limiti.

Un sentito ringraziamento va ai miei amici, vicini e lontani, con cui ho condiviso momenti indimenticabili e che sono sempre stati capaci di strapparmi un sorriso o darmi preziosi consigli.

Un grazie di cuore alla mia fidanzata, Ana, che con il suo affetto è riuscita a motivarmi e ha creduto in me anche quando io non ne ero in grado.

Infine ringrazio Carmen Marino, Andrea Del Principe e il resto del CRITS, l'ISISS A. Magarotto e tutti i segnanti che hanno contribuito e partecipato alle attività svolte durante questa tesi.

Abstract

Sign language is the primary means of communication for the Deaf and Hard of Hearing (DHH) community. As the production of digital media grows, relying solely on human interpreters is not always feasible, creating the need for automated, scalable solutions. Signing avatars offer a promising approach, but present significant challenges due to the linguistic and expressive complexity of sign languages. This thesis presents a prototype system that dynamically generates Italian Sign Language (LIS) sentences on a MetaHuman 3D avatar by smoothly blending individual signs recorded through motion capture. The system was developed at the Rai Centre for Research, Technological Innovation and Experimentation (CRITS), extending their existing platform for generating fixed sentences. My contribution enables users to construct custom sentences from a dictionary of signs that can be easily expanded, producing output evaluated as natural and intelligible by certified signers and DHH users. The work demonstrates the feasibility of modular sign synthesis using high-fidelity avatars, and outlines future directions including automatic text-to-gloss translation and the integration of non-manual features, paving the way toward an accessible, on-demand sign language translation system.

Contents

List of Figures

Introduction

Motivation

Sign languages are the main mode of communication for Deaf and Hard of Hearing (DHH) people, who constitute about 5% of the world's population according to the World Health Organization [14]. These languages are not derived from spoken languages: they evolved independently, with their own grammatical structures, lexicons, and cultural nuances. Many of them, including Italian Sign Language (LIS) [16] are recognized as fully fledged natural languages, not a mere visual translation of a particular spoken language. For many DHH individuals, sign language is therefore not only a communication tool but also a core part of their cultural identity. Despite the growing recognition of the importance of sign languages for inclusion, accessibility in digital media remains limited. Subtitles are still the most widespread form of support, but they cannot replace the richness and immediacy of sign language. Subtitles require literacy in the spoken language, do not convey the same rhythm or emphasis, and cannot capture the visual-spatial grammar of signing. As a result, subtitles often leave DHH viewers at a disadvantage when engaging with audiovisual content, especially when it is fast-paced, informal, or linguistically complex. Signed content created by human interpreters represents the most faithful solution, but it is costly and difficult to scale. Producing signed versions of large volumes of online video or broadcast material requires significant human resources, which are often unavailable. The scarcity of qualified signers further exacerbates this problem. These limitations highlight the need for automated approaches that can complement and extend human signing. Signing avatars emerge as a promising alternative. They can generate signed content on demand, work in real time, and maintain consistency across different contexts. Compared to human interpreters, avatars are infinitely scalable and can be customized in appearance, signing style, and can easily be integrated with digital platforms. At the same time, avatars face significant challenges: sign languages rely not only on hand shapes and movements, but also on facial expressions, gaze, body posture, and timing. Replicating these features convincingly is technically complex, and any deficiency can reduce

the intelligibility or acceptance of the generated signing. Research on signing avatars is growing and spans several directions. Some works propose end-to-end synthesis systems capable of producing entire signed sentences directly from textual or gloss input [25] [19] [12]. Other studies address specific technical components, such as refining motion editing techniques to smooth transitions between signs [23], improving facial animation to capture non-manual markers [24], or building synthetic datasets to facilitate large-scale training [5]. A parallel line of research investigates how the DHH community perceives these avatars: for instance, whether they are culturally appropriate, understandable, or acceptable in everyday use [18] [17]. Studies also highlight the importance of involving DHH individuals directly in the design and evaluation process, both to improve technical quality and to ensure that systems meet real needs [9] [6]. These contributions have significantly advanced the field, but important gaps remain. In particular, most systems rely on large pre-defined corpora of signed sentences or on direct text-to-sign translation pipelines. This limits flexibility: users cannot easily compose new sentences outside the predefined data, and system updates require expensive new recordings. There is still limited research into modular, dictionary-based approaches where individual sign animations can be dynamically combined to generate arbitrary sentences. Such systems could provide greater scalability, reusability, and adaptability to different contexts. The challenge, however, is to ensure that the output remains natural enough to be intelligible and acceptable to DHH users, despite being assembled from smaller units. It is therefore essential to further develop research in modular synthesis to create tools that can both broaden accessibility of digital content and enable smoother communication between DHH individuals and hearing people who do not know sign language.

Methodology and Structure

The objective of this thesis is to demonstrate the feasibility of a modular sign synthesis system, and to explore the factors that may support or hinder the acceptance of its output by DHH individuals. The work is situated in the context of Italian Sign Language (LIS), but the methodology is general enough to be adapted to other signed languages. To this end, a small dictionary of LIS signs was recorded using a motion capture suit with the help of a certified signer. In addition, a full sentence was recorded in a single take. This dual dataset enabled a direct comparison between two approaches: animating the sentence on a 3D avatar using the single-take recording, and dynamically reconstructing the same sentence by blending together the individual signs from the dictionary. The comparison allowed an evaluation of whether the modular

system can approximate the naturalness and intelligibility of continuous recorded signing. The prototype was implemented using a MetaHuman 3D avatar in Unreal Engine, chosen for its high-quality rigging and expressive potential. The system allows users to construct custom sentences by selecting items from a dictionary, and automatically blends the selected animations to produce fluid signing output. Evaluation was conducted with both certified signers and DHH individuals, who provided feedback on intelligibility, smoothness, and overall acceptability of the avatar's performance. The remainder of this thesis is organized as follows: Chapter 1 surveys the current research landscape and technologies in sign language synthesis, highlighting strengths, limitations, and gaps. Chapter 2 describes the methodology used to record, process, and blend sign animations, and details the design and implementation of the prototype system. Chapter 3 presents the results of comparative testing and discusses feedback from experts and DHH participants. Chapter 4 concludes the thesis, summarizing contributions and proposing directions for future work, such as automatic text-to-gloss translation and the integration of more advanced non-manual features.

Chapter 1

State of the art

1.1 Challenges of Sign Language synthesis

Sign languages are visual languages that use multiple channels in parallel to convey information, which can be broadly categorized in manual features (hand shape and movement) and non-manual features (facial expression, gaze, mouthing, body posture). They differ significantly from written and spoken languages as they evolved separately and have their own grammatical structures and lexicon. Many of their characteristics pose a major challenge for the automated processing of signed content, as detailed by Cooper, Holt and Bowden [7]:

- Multimodality: the use of different communication channels at the same time.
- Spatial grammar: the signed space is often used to "place" subjects to reference later. Directional verbs such as "give" or "phone" differ depending on who is the subject and the object. Positional signs such as "bruise" or "tattoo" act on specific parts of the body.
- Non-manual features: even micro expressions are key in determining the sign's meaning, for example some signs only differ by lip shape.
- Sign variation: instead of using two signs for "run quickly" the sign for "run" would be sped up. Iconicity also means that signs imitating the thing they represent can be altered to better conform to the specific one which is referred to.

These are just a subset of the features of sign languages, but they already highlight how complex and structurally different from written languages they are. Another major problem is the lack of a widespread standardized encoding

for signs, which paired with the scarcity of annotated sign content makes the use of machine learning approaches difficult. Despite all of this, the field has progressed significantly in the years, making communication more and more accessible for DHH people.

1.2 End-to-end synthesis systems

Several attempts to build translation systems from written language or gloss to sign language have been made. The work of Younes and Noussaima [25] showcases a web app where content can be created in four different sign languages. It is based on the Dicta-Sign dataset, a corpus of British, German, Greek and French sign language covering European travel, annotated in HamNoSys [8]. This was converted to SiGML notation thanks to the HamNoSys2SiGML open-source tool, facilitating integration with the JASigning animation system [15]. This tool is capable of generating single signs animations or constructed sentences, and also features idle animations. The avatar can also be rotated or zoomed to better examine the signs. Its user interface can be seen in Figure (1.1). Despite its flexibility with four languages and the ability to construct custom sentences directly on a web app, the avatar lacks expressiveness and when switching between signs in a sentence the animation abruptly cuts to the neutral position, interrupting the signing flow.

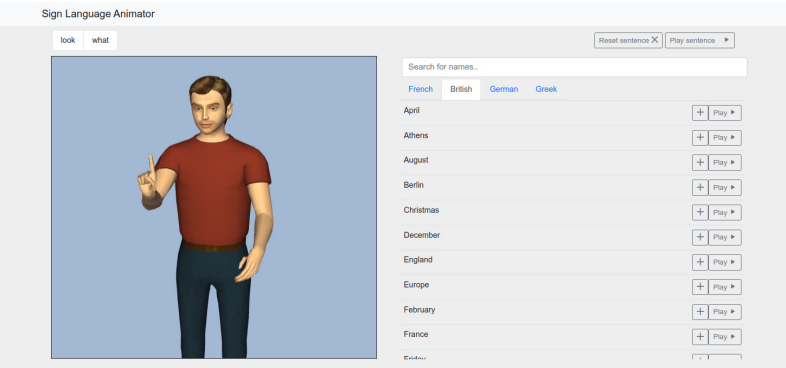

Figure 1.1. Sign Language Animator's user interface [25]

Another study by Gibet and Marteau [12] uses a novel hybrid approach, combining classic procedural synthesis with data-driven synthesis based on motion capture data. It also models the sign language aspects of spatiality and iconicity, allowing the generation of more inflected and spatially varied signs.

Finally, the work of Ribeiro, Dias, Faria and Romero [19] is also based on

motion capture. The mobile application is developed in Unity, and uses a state machine with an idle state and two alternating sign states in which glosses are cycled. To smooth the transitions the animations get overlapped and interpolated. This gave promising results which could be improved by refining accuracy and expressivity.

1.3 Recent technical improvements

Research has also focused on more specific aspects of sign language synthesis, trying to improve different parts of the pipeline. Uchida, Nakatani et al. [23] developed a motion editing tool that reduces the error rate in understanding Japanese Sign Language by adjusting the animations and integrating some of its grammatical elements (Figure 1.2). The tool is built in Unity, attaching to their preexisting motion blending pipeline. It allows the grouping of signs by manipulating their speed and blend span, which gives a more natural rhythm and clarifies sentence structure. It can also insert head nods while holding the hand position or pointing while holding the non-dominant hand.

Figure 1.2. JSL Motion Editing Tool features [23]

A study by Wolfe et al. [24] remarks how important facial features are in conveying both linguistic and paralinguistic information, and explores three themes that influence the current technology. Linguistic discovery, defining the facial activity that an avatar must carry out; Computer Generated Imagery (CGI), the foundation to build realistic avatars; and Sign Language Representation Systems, determining the fidelity of timing of facial co-occurrences.

Barros et al. [5] developed a large scale synthetic dataset for sign language with expressive signers. It includes additional ground truth data such as depth and normal maps, rendered models, segmentation masks and 2D/3D body joints. They also introduce a transformer-based GAN for realistic sign image synthesis. Together, they bridge the gap between real-world SL data scarcity and the high data needs of deep learning algorithms, offering a scalable path toward better SL production and accessibility tools.

1.4 Impact of DHH people involvement

Some recent studies have highlighted how important it is to involve DHH people more in the research and development of Sign Language processing technologies. Quandt et al. [18] ran a survey comparing three methods for signing: a live human signer, an avatar with computer-synthesized (CS) animation and an avatar with high fidelity motion capture (Figure 1.3). The results showed the mocap avatar was rated much more positively than the CS one, and that the signers who acquire sign language later in life are more accepting of and likely to have positive impressions of signing avatars. These findings suggest that the appearance and movement quality of an avatar are important for its acceptance.

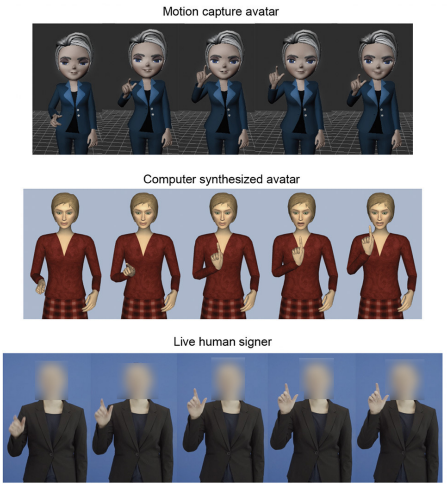

Figure 1.3. The three signers used in the study [18]

The work of Othman et al. [17] explored the acceptance of a culturally adapted avatar among DHH individuals in Qatar (Figure 1.4). While the results were generally positive, the participants viewed the signing avatar more as a tool to enhance accessibility than a substitute for human interpreters.

Figure 1.4. Culturally adapted avatar used in the study [17]

The research of Dimou et al. [9] proposes a methodology to involve the signers' community in the process of signing avatar development, via online surveys that allow the rating of various aspects of the synthetic signs (Figure 1.5).

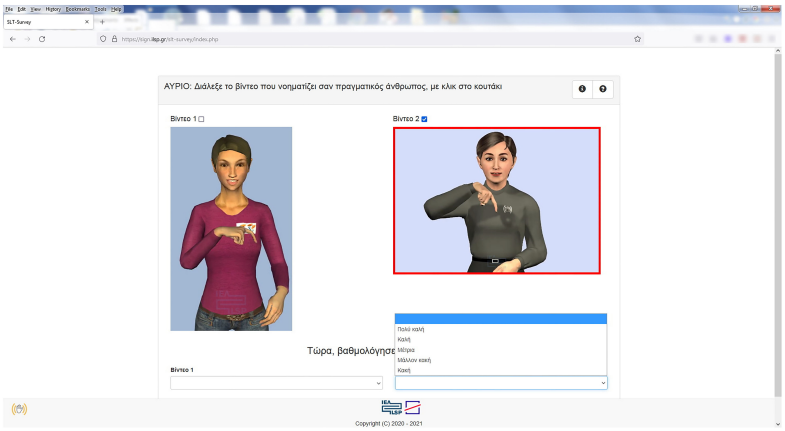

Figure 1.5. Comparison between different signers in online survey [9]

Finally, the study from Bragg et al. [6] summarizes the result of a three-day interdisciplinary workshop on sign language recognition, generation and translation. The focus is on the current state of Sign Language processing, the biggest challenges facing the field, and the possible calls to action. The identified calls to action are a greater involvement of Deaf team members, focus on real-world applications, developing UI guidelines for sign language systems, creating larger public video datasets, standardizing the annotation system and developing software for annotation support.

Overall, these studies highlight that technical advances alone are insufficient:

the meaningful involvement of DHH communities is essential for ensuring both usability and acceptance of sign synthesis systems.

None of these approaches is based on a modular sign synthesis that allows users to construct custom sentences from an easily expandable dictionary, seamlessly blending individual signs and producing output that is evaluated as natural and understandable by certified interpreters and DHH users thanks to high-fidelity avatars.

Chapter 2

Method

2.1 Recording signs with mocap

Motion capture, often shortened as "mocap" is the process of recording the movement of people or objects in a computer system with high precision. The captured information can then be mapped to a 3D model and used to animate it with CGI. The acquisition can be done in many ways, ranging from generic camera sensors paired with AI models to dedicated tracking suits with sophisticated sensors and it's widely used in cinema, video games, VR and robotics.

This technology is a good fit for recording sign language, due to the high sampling resolution and relative ease of use. In fact most modern setups have enough accuracy to properly reproduce the signs with naturalness and expressiveness, without being too uncomfortable for the signer.

The Rai Centre for Research, Technological Innovation and Experimentation (CRITS) has an Xsens commercial tracking suit, made up of unobtrusive inertial and magnetic sensors combined with advanced algorithms and biomechanical models [20]. Unlike camera-only approaches, this system provides reliable tracking even without a controlled studio environment. A pair of MANUS gloves is also used for enhanced hand and finger tracking accuracy, which is essential in the case of sign language. The setup is completed by an iPhone, which records the facial features. The gloves and suit are connected to a computer via dedicated wireless antennas, and managed through a proprietary Xsens software called "MVN Animate" (Figure 2.1). The facial features are recorded directly in Unreal Engine, thanks to the "Live Link" plugin which allows to stream the data from the iPhone's TrueDepth infrared sensor. The sampling rates are 60 FPS for the body and 24 FPS for the face.

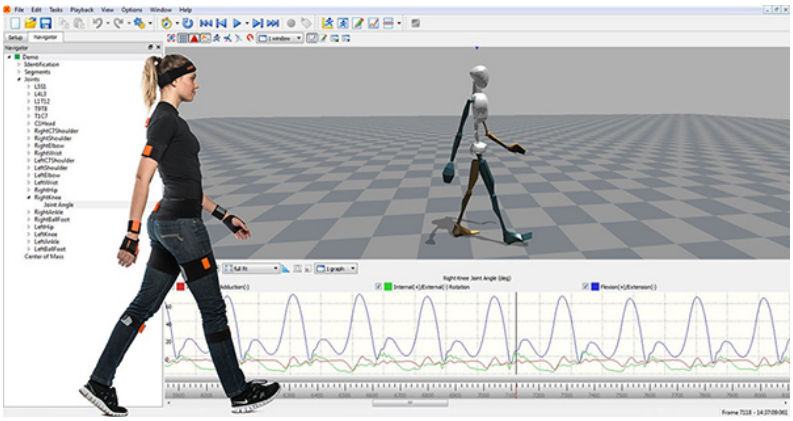

Figure 2.1. Xsens MVN Animate user interface

The first step when setting up the system is taking some measurements of the signer. This only needs to be done once, as this data can be stored for later use. The sensors can drift over time and electromagnetic interference can introduce noise, so the calibration process is essential to ensure precise acquisition results. First the signer must stand in N-pose, with the arms straight along the sides (Figure 2.2) or in T-pose, with the arms stretched horizontally. Then they must slowly walk in a circle so that all sensors are moving.

Figure 2.2. Calibration of suit in N-pose

Next, the gloves are calibrated by holding one hand still on a flat surface with the palm down, then repeatedly closing and opening the fingers, forming a fist (Figure 2.3). All of this is then repeated for the other hand.

Figure 2.3. Steps for glove calibration

The entire calibration process takes about 2 minutes, and can be repeated between takes if necessary. The software itself notifies the user if the sensor precision degrades. To begin the actual acquisition two separate recordings must be started at the same time: one on the Xsens MVN software for the body, and another one in Unreal for the face (Figure 2.4). The body animation is saved in a custom format, but it can be seamlessly converted in .fbx to import it in Unreal.

Figure 2.4. Recording setup in use

During the recording of the signs some issues came to light. Although non-critical, they introduced delays due to the needed manual intervention, and ignoring them would have impacted the final quality of the animations.

Initially the iPhone was held in place by a chest plate with shoulder straps, which had a protruding stick with a phone holder (Figure 2.5). This was impractical for several reasons: the stick was too low, occupying the signing space and making performing some signs difficult or impossible; this also made the avatar eyes look above where they should, and because the phone was attached to the chest it was not always perfectly lined up with the head, introducing errors in the tracking. This problem was solved by using a Rokoko Headrig, a lightweight head mount for phones specifically made for motion capture (Figure 2.6). Thanks to this helmet the face tracking became much more accurate and comfortable for the signer, although some breaks were still needed during capturing sessions.

Figure 2.5. Plate with phone holder

Figure 2.6. Rokoko Headrig [13]

Another problem was the sensitivity to electromagnetic interference. The CRITS laboratory where the recordings were conducted contains lots of networking equipment, in particular a wireless access point. This sometimes would cause errors in the communication between the suit's sensors and the radio receiver. The gloves especially suffered from this. For technical reasons the access points could not be moved from near the recording PC, but placing the radio receiver as far as possible from it and closer to the signer at least mitigated the issue. A few times the sensors would drift significantly. The solution in this case was repeating the calibration process between takes, to keep the inaccuracy as low as possible. Using the T-pose for calibration also seemed to improve the results.

2.2 Animating a MetaHuman in Unreal

MetaHumans are high-fidelity 3D avatars that combine realism, customization, and advanced rigging. As part of the Epic Games ecosystem, they integrate seamlessly with Unreal Engine (Figure 2.7). After choosing or creating your own MetaHuman in the MetaHuman creator web app, it can be imported into Unreal Engine from the Quixel Bridge window and then added to the scene.

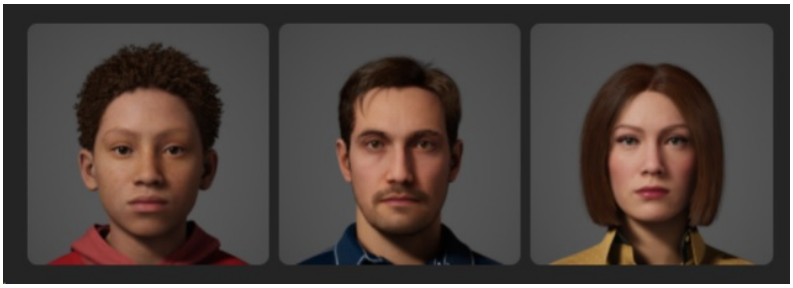

Figure 2.7. Metahumans created for the RAI-LIS project

Before importing the actual animations, a character with the same bone hierarchy and names as the MVN skeleton should be added to the Unreal project. This MVN Puppet (Figure 2.8) can be downloaded from the Xsens knowledge base, which includes a tutorial that explains in greater detail this procedure [11]. The recorded body animations can be imported by just dragging them into the content browser. In the prompt window, the aforementioned MVN Puppet must be selected as the skeleton.

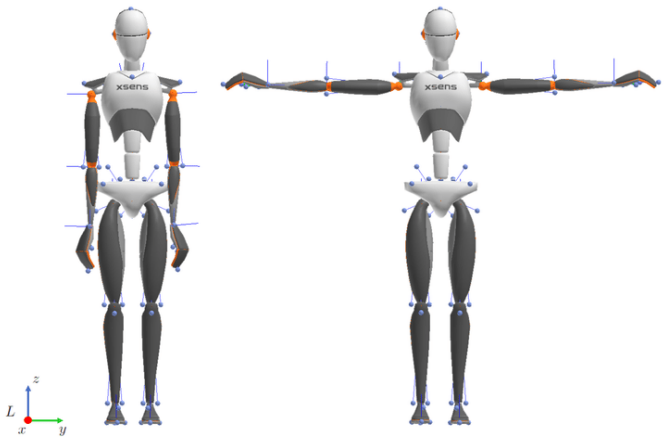

Figure 2.8. MVN skeleton in N-pose and T-pose

The MVN and MetaHuman skeletons differ in proportions and naming conventions, so direct playback would not align properly without retargeting. Retargeting is a process that maps animations from their original skeleton to another with a different structure. It can be a long manual process, but luckily from Unreal Engine 5.4 onward it can be done automatically [4]. A retarget asset can be generated and reused for the retargeting of subsequent animations that share the same source and destination skeletons. The retargeted animations can also be trimmed or adjusted by applying additive bone transformations in the Animation Sequence editor if needed, or directly in the Level Sequence later by baking it and adding a control rig with an additive track.

To reproduce the recorded animations on the MetaHuman a new Level Sequence must be created, and our character must be added to it. MetaHumans normally come with two control rig tracks for body and face live capture and manual editing. Since in this case the motion comes from pre-recorded animations, these default rigs must be removed to avoid conflicts, and replaced with animation tracks pointing to the imported MVN body clip and the face recording that was already in the project. The tracks must be synced manually to remove the offset between them, since the face capture is started after the body one (Figure 2.9). Once everything is setup this way, playing the level will start the animation and renders can also be done to create a video output.

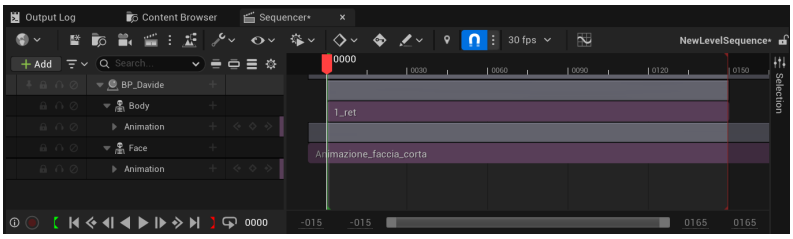

Figure 2.9. Sequencer timeline with the body and face animation synced

However, this pipeline only works for single-take recordings. If a mistake occurs or new signs are needed, the whole sequence must be re-recorded. To overcome this limitation, the next section explores how individual sign animations can be blended to form sentences dynamically.

2.3 Animation blending

2.3.1 First experiments

The first step before building an entire modular system was assessing if just blending two animations would produce a smooth transition between them. To this end a simple Level Sequence was created as described in the previous section, but a second animation was also added to the timeline, overlapping for a few frames with the first one (Figure 2.10). This allowed a quick test of the blending capabilities of the Unreal animation system. After tweaking the interpolation length a bit the output seemed smooth enough. Although the result was good, this method still required a lot of manual setup: for every sentence variation a timeline must be built ad-hoc, positioning and syncing all the tracks by hand. This makes the solution time consuming and definitely not scalable.

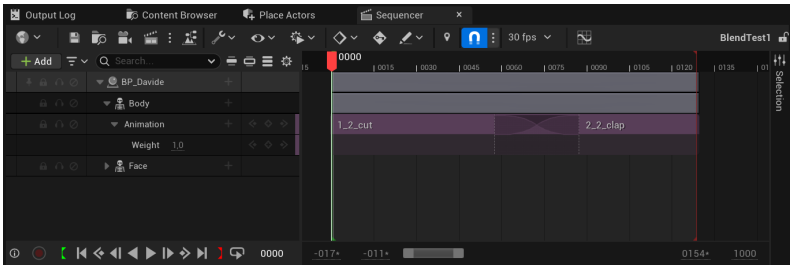

Figure 2.10. First manual blending attempt in the sequencer timeline

Another test was done with an Animation Montage asset, which seemed promising as it allows to define custom transitions between set animations, but it turned out even worse than the sequencer as it did not do any blending so the switch between signs was an abrupt cut.

Finally I opted for a state machine approach, as it offered a dynamic and scalable way to smoothly transition between signs: this method automatically blends the animation playing in a state with the one playing in the next, and the animations for each state can be set even at runtime. These features make it the perfect candidate to create a modular system that allows the dynamic production of full sign sentences composed from a dictionary of single signs.

At the core of this implementation there are two complementary states (A and B), which have the same internal logic: when one state gets near the end of its animation's playback, it sets the animation for the other state picking the next one from the animation queue and then triggers the transition. The process is repeated until the queue is emptied. This solution, although simple in concept required a lot of changes to the existing setup for it to work, which will

be detailed in the next sections.

2.3.2 MetaHuman blueprint changes

By default the MetaHuman blueprints have Actor as their parent class. "An Actor is any object that can be placed into a level, such as a Camera, static mesh, or player start location. Actors support 3D transformations such as translation, rotation, and scaling. They can be created (spawned) and destroyed through gameplay code (C++ or Blueprints)." [1] To allow using different MetaHumans while avoiding code duplication and keeping the logic centralized, a custom MetaHumanBase blueprint was created. This class inherits from Actor and contains the control functions for the playback, and all the related variables. To extend any MetaHuman character with these new functionalities all that is needed is changing their parent class to the custom MetaHumanBase.

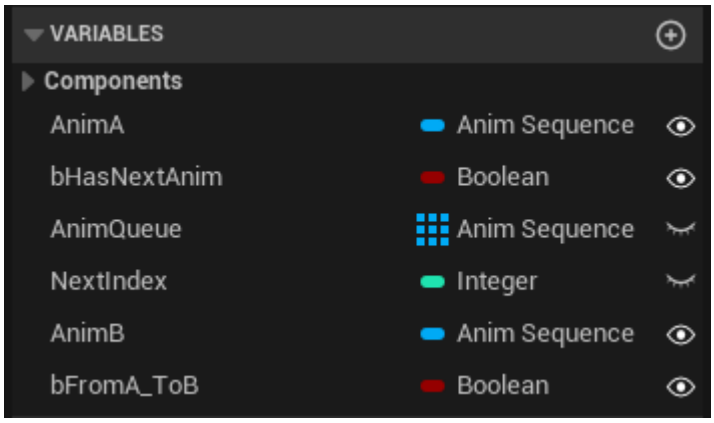

Figure 2.11. MetaHumanBase class variables

The variables, as can be seen in the figure 2.11 are

- AnimA/AnimB: references to the current Animation Sequence (the animation itself) to be played for each respective slot
- AnimQueue: Animation Sequence array storing all the signs that will be played continuously
- FromA_ToB: boolean keeping track of which state is active and which one needs to be set
- HasNextAnim: boolean to check if the queue is empty
- NextIndex: integer to keep track of the progress in the queue

Variables with the open eye icon are exposed, meaning they can be read from other classes too. This is fundamental for the correct functioning of the Animation Blueprint, which has local copies of these variable and updates them every frame. This separation between the MetaHumanBase and the Animation Blueprint is deliberate. The MetaHumanBase handles the high-level control logic (deciding which animation to play next, managing the queue, and updating the state flags). The Animation Blueprint on the other hand is only focused on executing those instructions: blending, transitioning, and rendering the animations. By keeping these responsibilities apart, the system remains modular, easier to debug and adaptable to different input sources.

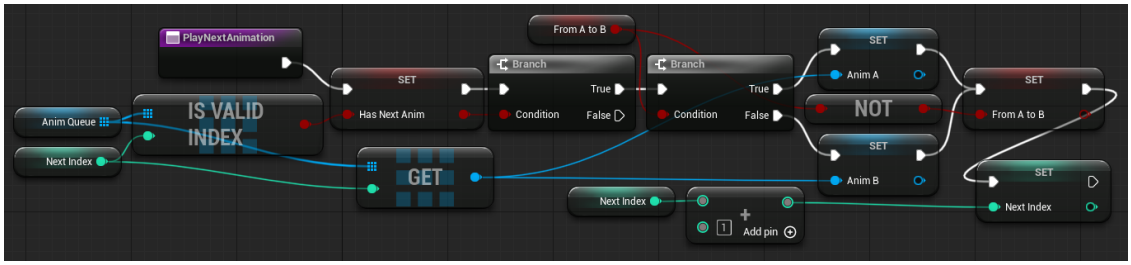

Figure 2.12. PlayNextAnimation function from the MetaHumanBase blueprint

The core of the MetaHumanBase blueprint is the PlayNextAnimation function (shown in Figure 2.12). First it checks that there is still an animation in the queue and updates HasNextAnim. If there is one it gets assigned to the AnimA or AnimB slot depending on the FromA_ToB flag, which then gets negated to update it for the next iteration. Finally the index counter gets incremented. This function is essential as it manages all the variables responsible for triggering and determining the conditions of the state transitions.

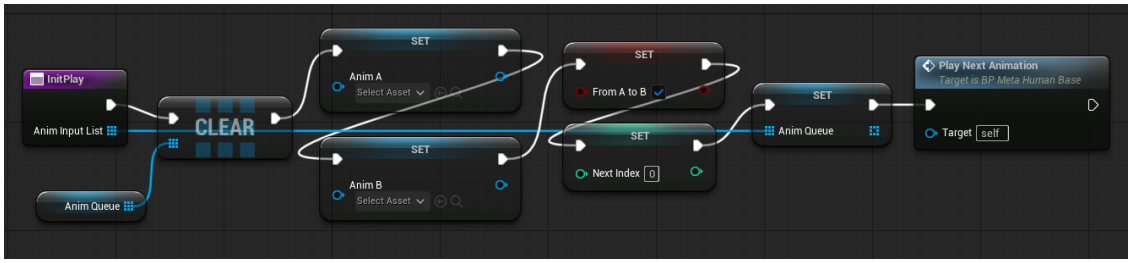

Figure 2.13. InitPlay function from the MetaHumanBase blueprint

The other function is InitPlay (Figure 2.13), which simply resets all the variables, sets the AnimQueue from the Animation Sequence array it receives as a parameter and calls PlayNextAnimation. The purpose of this function is to start the actual animation playback once the signs have been chosen.

This design makes the system reusable and keeps the animation control logic centralized, allowing different MetaHumans to be used interchangeably and the output to be customized for different contexts.

2.3.3 Animation blueprint and state machine

"Animation Blueprints are specialized Blueprints that control the animation of a Skeletal Mesh during simulation or gameplay. Graphs are edited inside of the Animation Blueprint Editor, where you can blend animation, control the bones of a Skeleton, or create logic that will define the final animation pose for a Skeletal Mesh to use per frame." [2]

This blueprint manages the actual animations of our MetaHuman characters. After creating the Animation Blueprint asset, it must be assigned to the Skeletal Mesh on the MetaHuman blueprint, by setting the animation mode on the Body component to "Use Animation Blueprint" and then setting the Anim Class to the Animation Blueprint asset that was just created, as seen in Figure 2.14.

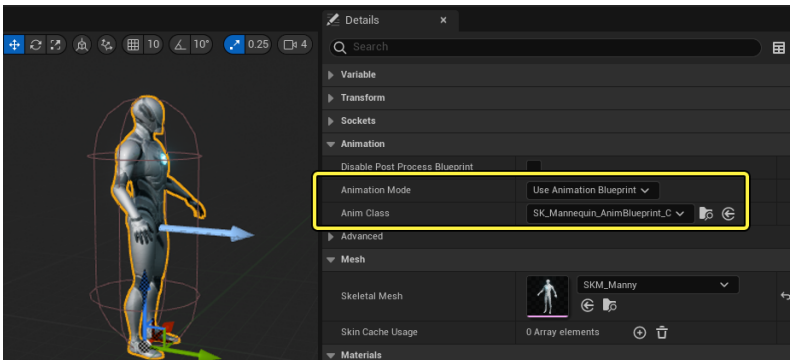

Figure 2.14. Assigning an Animation Blueprint asset to a Skeletal Mesh

The Animation Blueprint contains two main components: the EventGraph, which responds to specific events to manage animation variables and timing, and the AnimGraph, which contains the state machine definition. It also has a reference to the MetaHumanBase blueprint of the character it's assigned to, and local copies of the exposed variables of the MetaHuman (Figure 2.15).

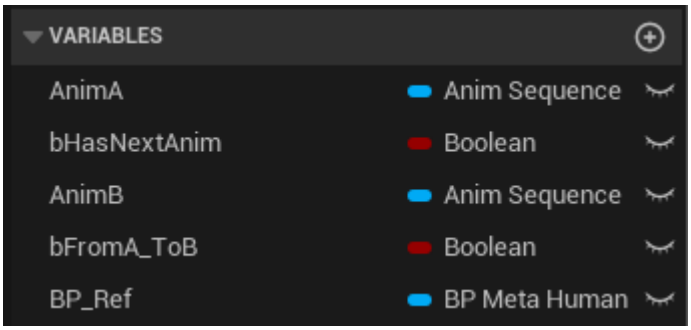

Figure 2.15. Animation Blueprint variables

"Events are nodes that are called from gameplay code to begin execution of an individual network within the EventGraph. They enable Blueprints to perform a series of actions in response to certain events that occur within the game, such as when the game starts, when a level resets, or when a player takes damage." [10]

The first captured event in the EventGraph is the "Blueprint Initialize Animation". This makes it possible to obtain a reference to the owning Actor of the Animation Blueprint, which in this case is the MetaHumanBase of the MetaHuman Skeletal Mesh it's assigned to. Storing this reference only once on blueprint initialization makes the code lighter, as the MetaHumanBase blueprint will be frequently accessed. This way only one call to GetOwningActor is needed, and the MetaHumanBase blueprint can be accessed directly with minimum overhead (Figure 2.16).

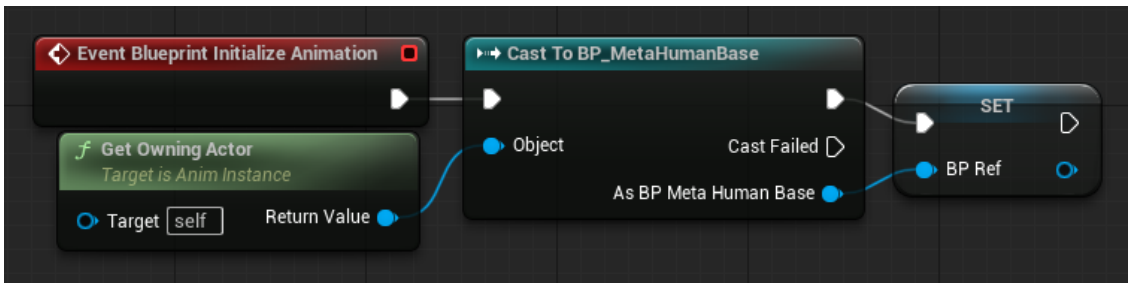

Figure 2.16. Event Blueprint Initialize Animation response

The "Blueprint Update Animation" event is called every frame, and it's used to update the local variables, reading the values from the exposed ones in the MetaHumanBase blueprint (Figure 2.17). This is essential to keep the

MetaHumanBase's control logic and the Animation Blueprint's state machine synchronized.

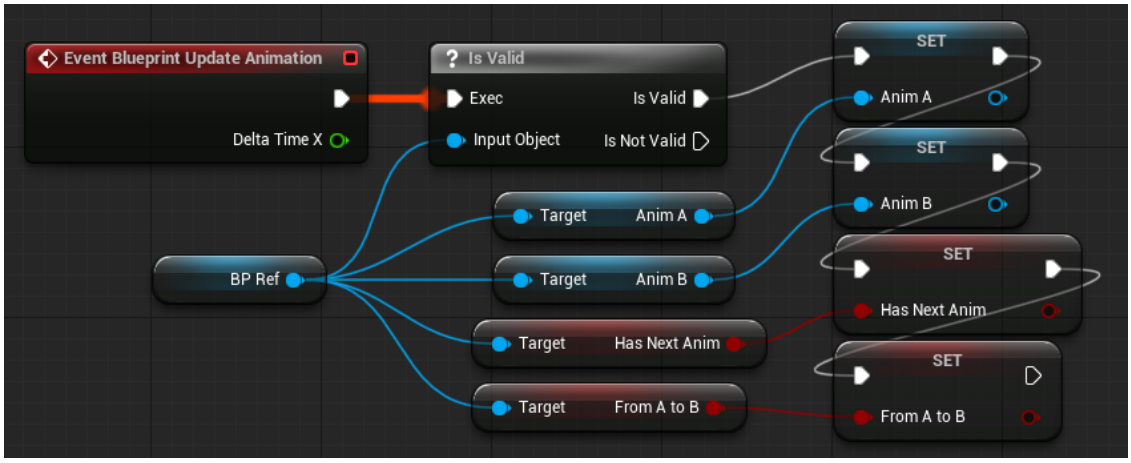

Figure 2.17. Event Blueprint Update Animation response

The last event is an "AnimDone" Animation Notify, a custom type of repeatable events synchronized to Animation Sequences [3]. These were used to signal the end of the sign inside the Animation Sequence. Once the custom notify is created it can be added to the animations by right clicking on their Notify track, and then it can be dragged to reposition it (Figure 2.18). This allows easy editing of the signs' timings in the dictionary at any moment without touching the rest of the system.

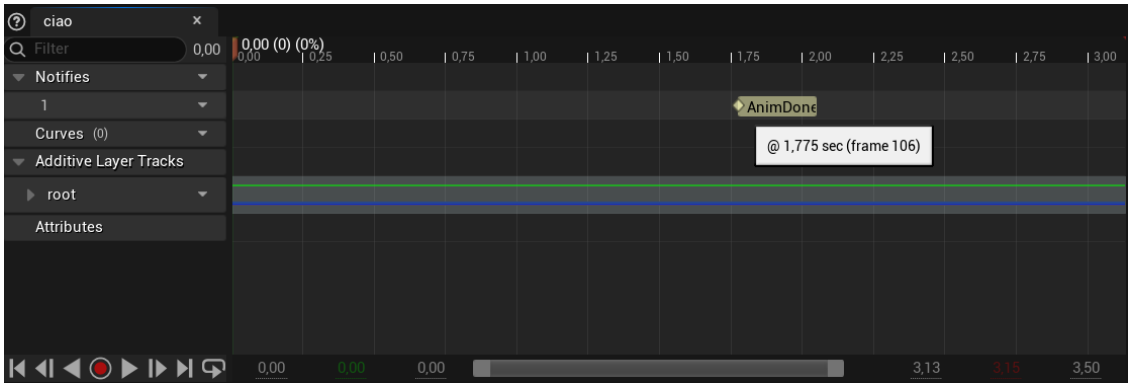

Figure 2.18. AnimDone Anim Notify in an Animation Sequence

When an AnimDone notify is launched, the Animation Blueprint calls the PlayNextAnimation function of the MetaHumanBase class, which advances the queue and updates the control flags, triggering the state transition. (Figure 2.19). For this reason, every Animation Sequence in the dictionary must include the AnimDone notify, otherwise playback would stop at the current sign without advancing.

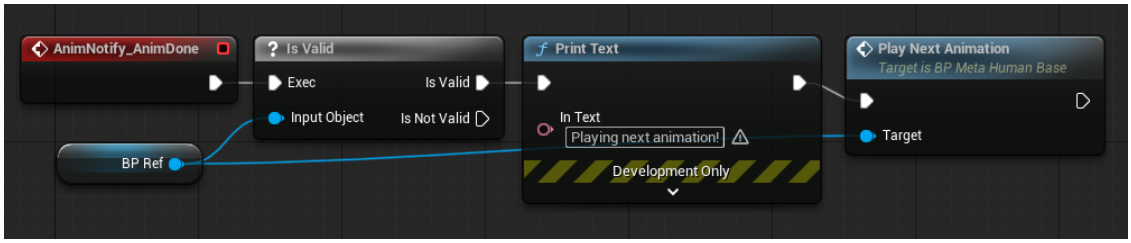

Figure 2.19. Anim Notify AnimDone response

"State Machines are modular systems you can build in Animation Blueprints in order to define certain animations that can play, and when they are allowed to play. Primarily, this type of system is used to correlate animations to movement states on your characters, such as idling, walking, running, and jumping. With State Machines, you will be able to create states, define animations to play in those states, and create various types of transitions to control when to switch to other states. This makes it easier to create complex animation blending without having to use an overly complicated Anim Graph." [21]

The state machine itself is defined inside the AnimGraph, and consists of three simple states (Figure 2.20): Idle, which plays a looping resting animation, PlayA and PlayB which play the Animation Sequences referenced in the AnimA and AnimB variables respectively (Figure 2.21).

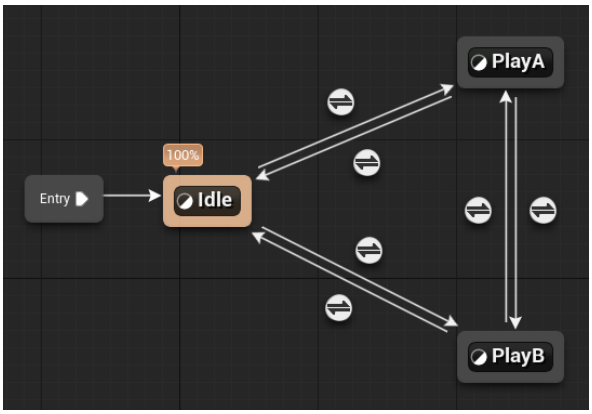

Figure 2.20. State machine diagram

Figure 2.21. PlayA state logic

The conditions for transitioning between states are determined by the HasNextAnim and FromA_ToB local booleans, and the state's Animation Sequence reference. Figures 2.22, 2.23 and 2.24 show the conditions of the PlayA state: in particular, for the Idle to PlayA transition AnimA must not be a null reference, HasNextAnim must be true and FromA_ToB must be false; for the PlayA to PlayB transition FromA_ToB and HasNextAnim must be true; for PlayA to Idle HasNextAnim must be false. The ones for the PlayB state follow the same logic but in a complementary way. In practice, this ensures that as long as a next sign is available, the system alternates between PlayA and PlayB, while returning to Idle when no sign is queued.

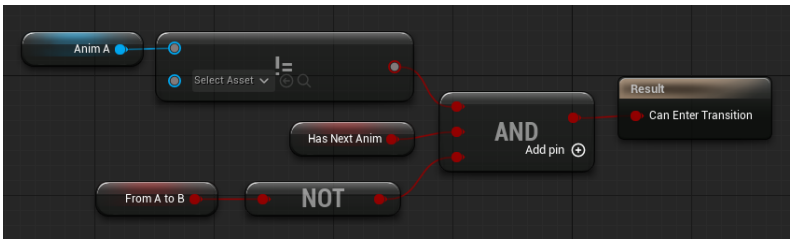

Figure 2.22. Idle to PlayA transition conditions

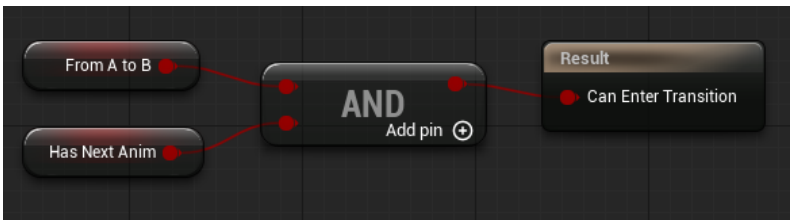

Figure 2.23. PlayA to PlayB transition conditions

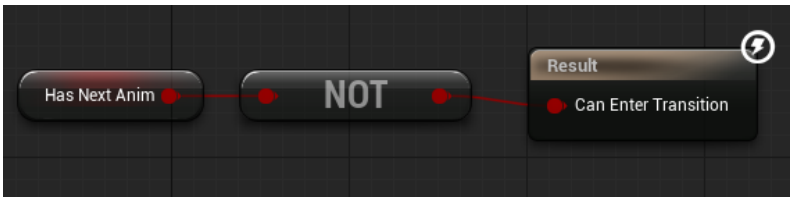

Figure 2.24. PlayA to Idle transition conditions

As mentioned before, the state machine in the Animation Blueprints performs automatic blending when transitioning between different states. This is essential to have a smooth playback without abrupt cuts, but to achieve pleasant looking transitions the right combination of blend curve and duration must be chosen. After experimenting with different options, the Hermite-Cubic InOut curve with a 0.5 second blend duration provided the most natural transitions between signs. This curve avoids the robotic stiffness of linear blending while preventing the overshoot of smoother but slower curves.

In summary, the Animation Blueprint functions as the execution layer of the system, synchronizing with the MetaHumanBase blueprint and ensuring smooth sign playback through its state machine and blending logic.

2.3.4 User interface

At this point the system is functionally complete, but the InitPlay must be called with a list of Animation Sequences for anything to happen. For testing and developing purposes this was done from the BeginPlay Event response in the Level Blueprint, which was launched when the level starts playing, and a static array of preset animations was passed. But to easily select, queue and playback different sign animations in real time a simple user interface was needed. This would also give a better user experience that allows to appreciate the modularity of this dictionary-based system.

Three different MetaHumans were placed on the level to showcase the prototype's flexibility. The UI consists of a list of buttons on the left side of the screen, displaying the available signs. Clicking a button adds that sign to a second list on the right, which shows the queued sequence of signs. Signs can also be removed by clicking on them directly in the queue. Once the sentence is complete, playback can be started with a button in the top-right corner. (Figure 2.25)

Figure 2.25. User interface showing the available and queued signs

This UI was developed with Unreal Motion Graphics (UMG). The main component, a Widget Blueprint, contains two List Views (one for the available signs and one for the chosen signs), a button to initiate playback and a reference to the MetaHumanBase targets. On initialization all the MetaHumans on the level are added as targets, and the available signs list gets populated from a Data Table containing display names and references for the Animation Sequences. Adding and removing signs from the queue and starting the playback are handled by simple click events.

The user interface was developed as a testing tool for this prototype. It is not designed for direct use by DHH users, but rather to provide a convenient way to interact with the animation system during development.

2.3.5 Creating the sign dictionary

The beginning of this chapter illustrated the general procedure to record animations with a motion capture setup and reproduce them on a character in Unreal. But to build the sign dictionary it is important to keep in consideration what pose to use as starting and ending position for each sign, as this is a design decision that will greatly influence the quality of the blending between them. For this prototype two main batches of signs were recorded with a certified signer. In the first batch, the N-pose was used as resting position as it appeared the most natural (Figure 2.26).

Figure 2.26. N-pose as resting position between signs

This first experiment revealed some limitations: sometimes weird effects were produced during the transitions, as moving the arms all the way down the sides took a significant amount of time and space. For this reason during the second batch of recording a different resting position was used, with the hands clasped in front of the chest (Figure 2.27). This significantly improved the smoothness of the transitions, as the hands moved less and remained closer to the signing space. Incidentally, this also made sense logically as the resting pose corresponded to the sign for ending a period in LIS.

Figure 2.27. Resting position with the hands closer to the signing space

To allow a direct comparison between continuous and modular playback, the sentence "Ciao, benvenuti al centro ricerca RAI." was recorded both in a single take and as individual signs. The latter recordings were then added to the sign dictionary, making it possible to reconstruct the sentence by sequencing its components. Some other signs were later added for further testing.

2.3.6 Generating video output for evaluation

To generate the signed clips needed for the evaluation survey the Take Recorder was used. This tool allows capturing animation directly during gameplay, or from live performances and other sources [22]. After adding one of the MetaHumans and a CineCameraActor as sources in the Take Recorder the level was played, displaying the prototype's UI. Once the sentence was constructed, the take recording was started and then the playback launched (Figure 2.28).

Figure 2.28. Capturing the blended animations with the Take Recorder

This produced a Level Sequence with the MetaHuman and camera animation tracks, which was then rendered with the Movie Render Queue. The output was single image frames, which were then merged in a final video file with an external program. The same procedure was then repeated with the single Animation Sequence containing the one-take sentence to have a comparison baseline. The resulting videos from the continuous recording and the reconstructed sentence were later presented side-by-side in the survey to assess the intelligibility and naturalness of the modular approach.

Chapter 3

Results

3.1 First observations

During one of the recording sessions the yet unfinished prototype was shown to the certified signer. In this phase only the first batch of signs was available (the ones with the N-pose neutral position) and the UI did not exist. Regardless, the certified signer gave some very positive feedback, noting that the transitions were smooth and natural-looking, and the individual signs were accurate enough to correctly interpret their meaning. The signer also reported some jerkiness in the beginning and end of the playback, which was promptly fixed by adjusting the blending times. This early feedback, while anecdotal, was valuable in highlighting both the strengths and weaknesses of the system from the perspective of a domain expert.

Another quick evaluation was done by comparing the developed prototype to the preexisting RAI system for signed content generation. This consists of a web app with a dictionary of sign animations that can be composed and played by an avatar (Figure 3.1).

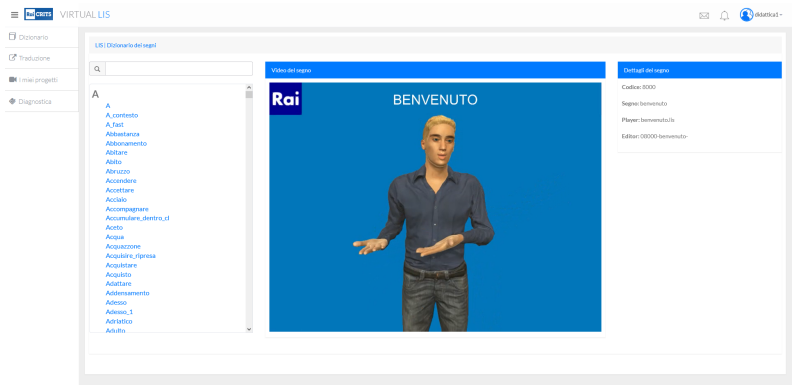

Figure 3.1. Web app for the old RAI signing system

Several key improvements emerge when comparing the two systems. First of all the avatar itself looks much more detailed and realistic, drastically reducing the uncanny valley effect (Figure 3.2). This is also exacerbated by the fact that the animations for the old system were created by hand. Besides making the movements look rigid and robotic, this manual approach makes the system hardly scalable despite it being modular too: the current dictionary is composed of around a thousand signs, and building a similar sized database with the motion capture approach would require significantly less time while providing natural looking animations that fully capture the expressiveness of the signer.

Figure 3.2. Comparison of the old and new signing avatars

3.2 Evaluation survey

To complement the preliminary observations, a formal evaluation was conducted through a structured survey aimed at assessing both the base motion capture setup and the effectiveness of the modular blending system.

The evaluation survey is comprised of two parallel sections: in the first one a video of the LIS interpreter signing the full "Ciao, benvenuti al centro ricerca RAI." sentence is shown, next to a video of the avatar reproducing the same continuous animation. This gives us both an assessment of the quality and accuracy of the mocap and avatar setup by themselves, and a baseline to compare to the blending prototype to judge its efficacy.

In the second section there is a video of the LIS interpreter performing the same signs individually, pausing and going back to the clasped hands neutral pose between each of them and then next to it there is the video of the avatar playing the sentence reconstructed from the individual signs. This section serves to isolate the blending system's contribution and highlight how well it works and if it is a viable option.

Both sections feature a set of six Likert items with a rating from 1 (strongly disagree) to 5 (strongly agree) to evaluate different parameters, and an optional text field for further observations. The items are as follows:

- The signs produced by the avatar are easy to understand.
- The avatar faithfully reproduces the signs shown by the interpreter.
- The transitions between the avatar's signs are smooth and fluid.
- The movements of the avatar appear natural.
- The overall visual quality of the avatar's signs is adequate.
- I would use or recommend this avatar in communication activities.

The survey was completed by 13 participants, including both certified LIS signers and DHH people.

The intelligibility evaluation was generally positive, and it remained consistent between the single take recording and the reconstructed sentence. (Figure 3.3) This was expected as it mainly depends on the recording setup and the avatar. Most participants rated both versions above the neutral midpoint, confirming that the mocap-driven animations convey meaning effectively even without additional cues. The use of motion capture allowed recording of movements with high spatial and time resolution, and the advanced rigging capabilities of the MetaHumans allow to properly reproduce that.

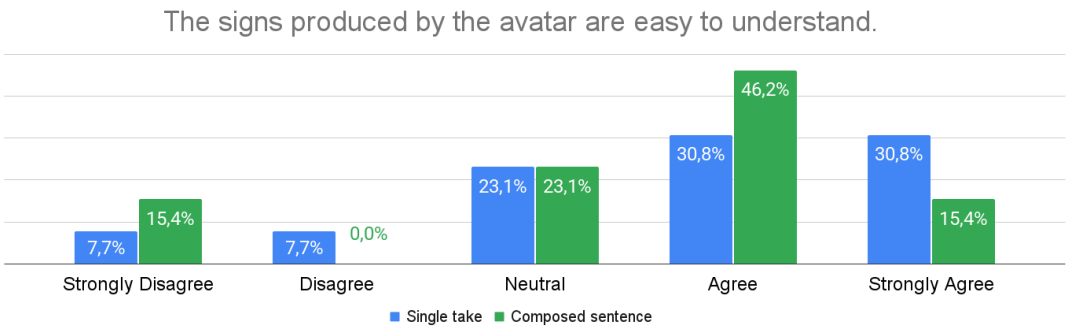

Figure 3.3. Survey results for intelligibility

Fidelity results followed a similar trend, reflecting the same dependence on the underlying mocap and rigging setup. Minor differences between the single-take and reconstructed versions suggest that the blending system preserves most of the signers motion characteristics. (Figure 3.4)

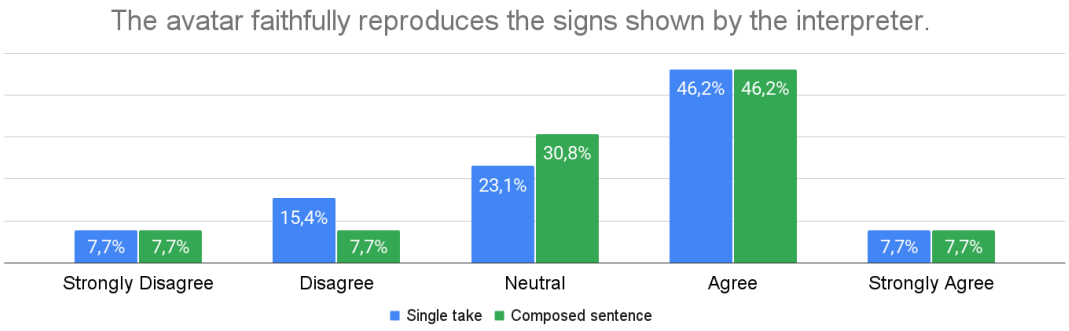

Figure 3.4. Survey results for fidelity

Blending smoothness results were somewhat surprising: while in both cases the majority of participants gave a positive score, I was expecting much higher results for the single take playback since it doesn't have any blended transition, instead reproducing the sentence in full as originally signed by the interpreter. These results indicate that the proposed blending approach is functional and yields satisfactory transitions. (Figure 3.5)

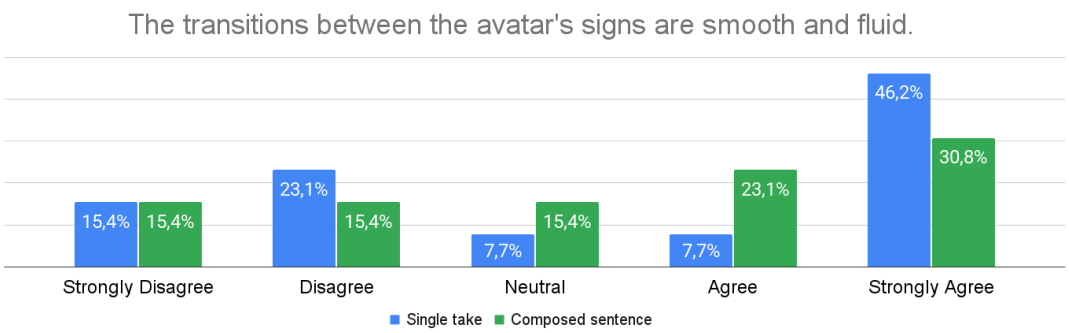

Figure 3.5. Survey results for transition smoothness

The naturalness scores are also a bit unexpected, as they are mixed in both cases, but get slightly better in the composed version instead of worse as I would have thought. (Figure 3.6)

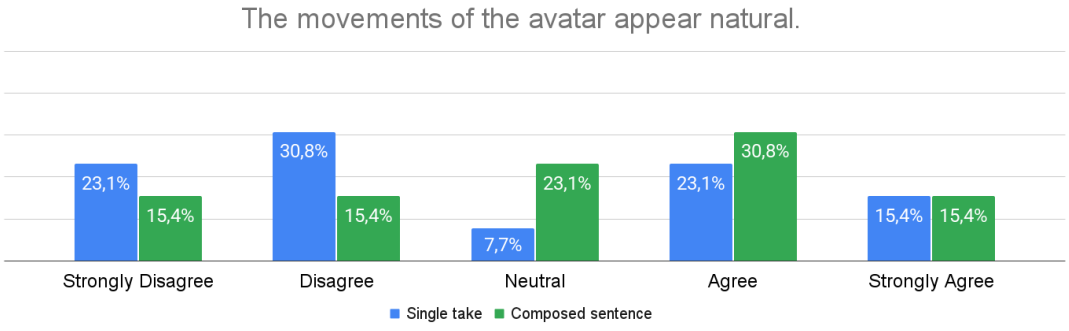

Figure 3.6. Survey results for naturalness

The overall visual quality was also positively rated, with minor differences between the two versions. (Figure 3.7)

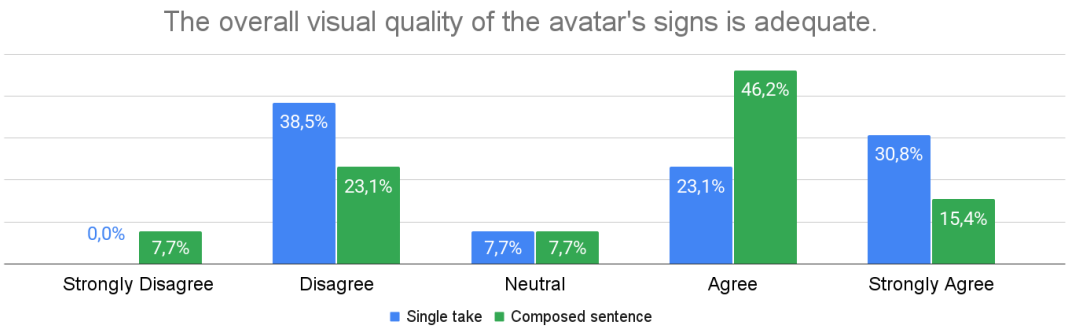

Figure 3.7. Survey results for overall quality

Results for acceptability are mixed, but positive on average. In this case too the composed sentence got slightly better scores. (Figure 3.8)

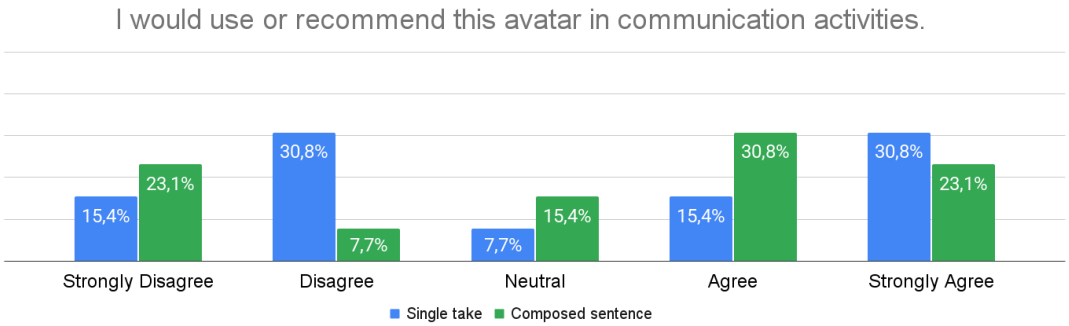

Figure 3.8. Survey results for acceptability and usability

The optional open questions gathered more useful insights from the participants. Many praised the intent and recognized the potential, but noted that it needs more polishing, especially with regards to expressiveness. These comments align with the broader observation that expressiveness, particularly through non-manual features, remains the most prominent area for improvement.

Overall, the survey results indicate that the modular blending approach achieves a level of intelligibility and naturalness comparable to continuous recordings, while maintaining flexibility and scalability. Although some respondents noted limitations in expressiveness, the positive ratings across most criteria confirm the viability of the system as a foundation for further development.

Chapter 4

Conclusion

4.1 Conclusion

Sign language synthesis remains a key challenge in making digital communication more accessible to the Deaf and Hard of Hearing (DHH) community. This thesis contributes to that ongoing effort by presenting a modular sign language synthesis tool that enables the construction of custom sentences built from a dictionary of pre-recorded signs, and their playback on a high-fidelity avatar.

After an overview of the current landscape in sign language synthesis, including the challenges that make it a difficult technology to develop and adopt, the details of the prototype and its implementation are described: the motion capture setup, the recording process, the playback method are introduced first, as they serve as the backbone for the blending system, which is the core of the work. Unlike many existing systems that rely on predefined sentence animations or low-fidelity models, this prototype demonstrates that modular blending of recorded signs can achieve natural motion with relatively low complexity.

The prototype was then evaluated through a user survey. The results are generally positive, showing that the prototype is successful in conveying meaning while retaining naturalness. The blending system obtained a score of 53.9% (strongly agree and agree) compared to the same percentage for continuous signals, demonstrating the effectiveness of the solution implemented. Furthermore, the overall visual quality score was 61.6% (strongly agree and agree), demonstrating the high quality of the system implemented. Some testers noted there is still a lack of expressiveness, which might be due to the missing facial animations. These findings suggest that while intelligibility and fluidity can be achieved through modular blending, perceived expressiveness remains tightly coupled to non-manual cues, an area that warrants further exploration. Nevertheless, most participants rated the output as useful for communication activities, and the

quality and acceptance rate could be further improved with targeted adjustments.

4.2 Future works

This work expanded the current RAI system for signed content generation, making it more flexible and reducing the need to record entire sentences with an interpreter every time. The system can be further improved with other additions, notably:

- Facial features: this is an essential part of sign language, adding them would greatly enhance the expressiveness of the avatar and make it more acceptable by DHH people.
- Full translation pipeline: adding a translation layer from written or spoken language to gloss would make this system not only useful for signed content generation, but also for improved accessible communication in everyday scenarios, eventually even in real time.
- Dataset scaling: capturing a larger dictionary to improve linguistic coverage and reduce repetition.
- Sign Language grammar modeling: implementing sign modifiers or the ability to combine different ones will dramatically improve the system's communicative potential without needing to record all sign variations and combination.
- Artificial Intelligence: exploring the use of large language models. With their remarkable reasoning abilities and rich knowledge, they have revolutionized many tasks, although their impact on sign language generation remains limited due to its complexity and unique rules.

Ultimately, this project demonstrates that modular synthesis using motion-captured signs can serve as a practical foundation for accessible and scalable sign language technologies, contributing to the broader goal of inclusive digital communication.

Bibliography

[1] *Actors in Unreal Engine | Epic Developer Community — dev.epicgames.com.* https://dev.epicgames.com/documentation/en-us/unreal-engine/actors-in-unreal-engine.

[2] *Animation Blueprints in Unreal Engine | Epic Developer Community — dev.epicgames.com.* https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-blueprints-in-unreal-engine.

[3] *Animation Notifies in Unreal Engine | Epic Developer Community — dev.epicgames.com.* https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-notifies-in-unreal-engine.

[4] *Auto Retargeting in Unreal Engine | Epic Developer Community — dev.epicgames.com.* https://dev.epicgames.com/documentation/en-us/unreal-engine/auto-retargeting-in-unreal-engine.

[5] Jilliam M. Díaz Barros et al. "SynthSL: Expressive Humans for Sign Language Image Synthesis." In: *2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG)*. 2024, pp. 1–10. DOI: 10.1109/FG59268.2024.10582038.

[6] Danielle Bragg et al. "Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective." In: *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*. ASSETS '19. Pittsburgh, PA, USA: Association for Computing Machinery, 2019, pp. 1631. ISBN: 9781450366762. DOI: 10.1145/3308561.3353774. URL: https://doi.org/10.1145/3308561.3353774.

[7] Helen Cooper, Brian Holt, and Richard Bowden. "Sign Language Recognition." In: *Visual Analysis of Humans: Looking at People*. Ed. by Thomas B. Moeslund et al. London: Springer London, 2011, pp. 539–562. ISBN: 978-0-85729-997-0. DOI: 10.1007/978-0-85729-997-0_27. URL: https://doi.org/10.1007/978-0-85729-997-0_27.

[8] *Dicta-Sign Corpus — sign-lang.uni-hamburg.de*. https://www.sign-lang.uni-hamburg.de/lr/compendium/corpus/dictasign.html.

[9] AthanasiaLida Dimou et al. "What about synthetic signing? A methodology for signer involvement in the development of avatar technology with generative capacity." In: *Frontiers in Communication* 7.798644 (2022). DOI: 10.3389/fcomm.2022.798644. URL: https://doi.org/10.3389/fcomm.2022.798644.

[10] *Events in Unreal Engine | Epic Developer Community — dev.epicgames.com*. https://dev.epicgames.com/documentation/en-us/unreal-engine/events-in-unreal-engine.

[11] *FBX import into Unreal Engine — base.movella.com*. https://base.movella.com/s/article/FBX-import-into-Unreal-Engine?language=en_US.

[12] Sylvie Gibet and Pierre-François Marteau. "A Text-To-SL Synthesis System Using 3D Avatar Technology." In: *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. 2023, pp. 1–5. DOI: 10.1109/ICASSPW59220.2023.10193734.

[13] *Headrig - A professional head mount for face capture by Rokoko — rokoko.com*. https://www.rokoko.com/products/headrig.

[14] Zambian Ministry of Health / Rachel Hapunda. *Deafness and hearing loss — who.int*. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

[15] *JASigning - Virtual Humans — vh.cmp.uea.ac.uk*. https://vh.cmp.uea.ac.uk/index.php/JASigning.

[16] *Misure per il riconoscimento della lingua dei segni italiana e l'inclusione delle persone con disabilita' uditiva*. https://www.gazzettaufficiale.it/atto/serie_generale/caricaArticolo?art.versione=1&art.idGruppo=5&art.flagTipoArticolo=0&art.codiceRedazionale=21A03181&art.idArticolo=34&art.idSottoArticolo=3&art.idSottoArticolo1=10&art.dataPubblicazioneGazzetta=2021-05-21&art.progressivo=0.

[17] Achraf Othman et al. "The Acceptance of Culturally Adapted Signing Avatars Among Deaf and Hard-of-Hearing Individuals." In: *IEEE Access* 12 (2024), pp. 78624–78640. DOI: 10.1109/ACCESS.2024.3407128.

[18] Lorna C. Quandt et al. "Attitudes Toward Signing Avatars Vary Depending on Hearing Status, Age of Signed Language Acquisition, and Avatar Type." In: *Frontiers in Psychology* 13 (2022). DOI: 10.3389/fpsyg.2022.730917.

[19] Bruno Ribeiro et al. "Capturing and Processing Sign Animations to a Portuguese Sign Language 3D Avatar." In: *2023 30th International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2023, pp. 1–5. DOI: 10.1109/IWSSIP58668.2023.10180233.

[20] Martin Schepers, Matteo Giuberti, and G. Bellusci. *Xsens MVN: Consistent Tracking of Human Motion Using Inertial Sensing*. Mar. 2018. DOI: 10.13140/RG.2.2.22099.07205.

[21] *State Machines in Unreal Engine | Epic Developer Community — dev.epicgames.com*. https://dev.epicgames.com/documentation/en-us/unreal-engine/state-machines-in-unreal-engine.

[22] *Take Recorder in Unreal Engine | Epic Developer Community — dev.epicgames.com*. https://dev.epicgames.com/documentation/en-us/unreal-engine/take-recorder-in-unreal-engine.

[23] Tsubasa Uchida et al. "Motion Editing Tool for Reproducing Grammatical Elements of Japanese Sign Language Avatar Animation." In: *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. 2023, pp. 1–5. DOI: 10.1109/ICASSPW59220.2023.10193198.

[24] Rosalee Wolfe et al. "State of the Art and Future Challenges of the Portrayal of Facial Nonmanual Signals by Signing Avatar." In: July 2021, pp. 639–655. ISBN: 978-3-030-78091-3. DOI: 10.1007/978-3-030-78092-0_45.

[25] Ouargani Younes and El Khattabi Noussaima. "Sign Language Animator: A Platform for Multilingual Sign Language Synthesis." In: *2024 International Conference on Intelligent Systems and Computer Vision (ISCV)*. 2024, pp. 1–8. DOI: 10.1109/ISCV60512.2024.10620089.