This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D2.1 Sign language animation preliminary development and production

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.:

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence. – IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	CERTH
INVOLVED UNITS:	CERTH
DOCUMENT NUMBER:	D2.1
DOCUMENT TITLE:	Sign language animation preliminary development and production
WORK-PACKAGE:	WP2
DELIVERABLE TYPE:	Demonstrator
CONTRACTUAL DATE OF DELIVERY:	30-09-2018
LAST UPDATE:	30-09-2018
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
v.0.1	16/08/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Dimitrios Tzovaras (CERTH/ITI)	Table of Contents definition and document structure
v.0.2	21/08/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 1, Chapter 2
v.0.3	24/08/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 3
v.0.4	29/08/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 4
v.0.5	07/09/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 5
v.0.6	17/09/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 6.1 , 6.2
v.0.7	17/09/2018	Draft	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 6.3
v.0.8	26/09/2018	Version ready for peer-review	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI)	Chapter 7, Chapter 8

			Dimitrios Tzovaras (CERTH/ITI)	
v.1.0	30/09/2018	Final Version	Nikolaos Kaklanis (CERTH/ITI) Dimosthenis Elmas (CERTH/ITI) Konstantinos Votis (CERTH/ITI) Georgios Gerovasilis (CERTH/ITI) Dimitrios Tzovaras (CERTH/ITI)	Final version with peer-review comments addressed

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
2D	2 Dimensional
3D	3 Dimensional
ASL	American Sign Language
DoW	Description of Work
FK	Forward Kinematics
fps	frames per second
IK	Inverse Kinematics
MoCap	Motion Capture

Table of Contents

List of Figures

Executive Summary

This document is the preliminary version of the deliverable D2.1 concerning the development and production of the EasyTV sign language animation method. This method will animate signer avatar focusing on realism. A signer avatar will receive the recorded sign language gestures and simulated in realistic way through the joint animation of both face and hand motions. In this first version, the requirements and the architecture of the module are outlined. Also, an analysis of the phases that compose the module is given, regarding the creation and development of avatar and the connection with crowdsourcing platform.

1. INTRODUCTION

The implementation of sign language service on the EasyTV project aims to visualize the data which have been obtained from the crowdsourcing platform, using an animated avatar. The service will animate signer avatar with excellent accuracy and realism in order to increase sign comprehesion. The development of the avatar will follow specific guidelines. The process of the reproduction of the recorded signed language will focus on synchronization of the joint animation of both face and hand motions and their embeddings into graph topologies. The product of this service will be ready to used ondifferent devices such as Desktop PCs (equipped with a web-browser) or an Android TV.

The present document is organised as follows:

- **Chapter 2** presents the architecture of the sign language 3D reproduction system conforming to the EasyTV Description of Work (DoW).
- **Chapter 3** gives an overview of the avatar technologies and describes the steps for creating a basic avatar
- **Chapter 4** describes the rigging methodology of the avatar's body and Inverse Kinematics.
- **Chapter 5** examines the facial expressions with emphasis in lip sync methodology.
- **Chapter 6** analyzes the most important part of sign language which is hand (and finger) gestures and the conversion in the avatar.
- **Chapter 7** describes the connection with crowdsourcing platform
- **Chapter 8** outlines the conclusions of this work and discusses future work.

2. SIGN LANGUAGE 3D REPRODUCTION SYSTEM ARCHITECTURE

This Chapter presents the aspects concerning the architecture of the EasyTV signer avatar service. These include the interconnection with other EasyTV services (i.e., external architecture), as well as, the internal components that compose the avatar development (i.e., internal architecture).

2.1. Input & output

In document D1.3 "*First release of the EasyTV system architecture*" the architecture of the Sign Language Production Module is defined. The module consists of three submodules, a) sign language crowdsourcing platform, b) capturing module and c) the realistic avatar. The complete architecture including interactions with other components is shown in Figure 2-1.

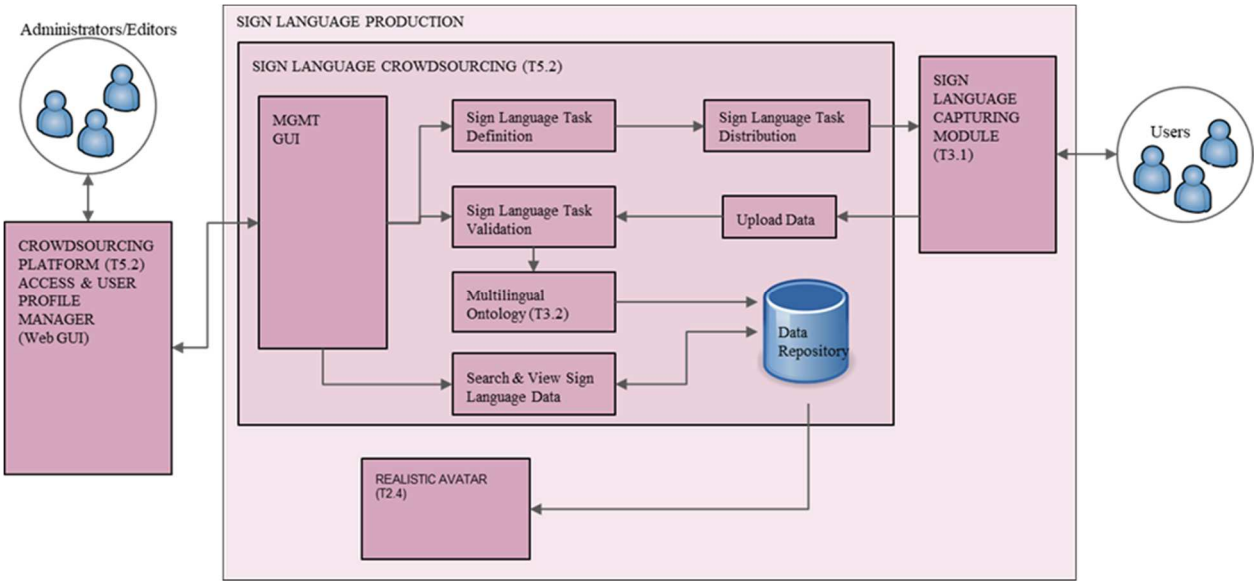

Figure 2-1: EasyTV Sign Language Production architecture

In general terms, the crowdsourcing module can be described as the manager of the data, capturing module production of new data and avatar for visualizing the data. Therefore, data repository can described as the final output of a procedure that includes creation and editing data. More specifically, the output of the capture module is a collection of JSON files that are uploaded and stored into repositories to the crowdsourcing platform after a bit of optimization and categorization. The content of these files - recorded signs represented from a signer, regarding face, body and hands motions – will be in their turn the main input for reproduction system architecture. The purpose is to convert the recorded signs to human moves from avatar representing the signer. A humanoid avatar with the appropriate rigging it's a precondition to reproduce the exact same moves. As it will be described later in this document, improvements will be made to avatar's movements focusing on realism in order to offer a better experience to the user. The output of the 3D reproduction system will be an avatar signer visualizing the recorder sign, stored in crowdsourcing repository.

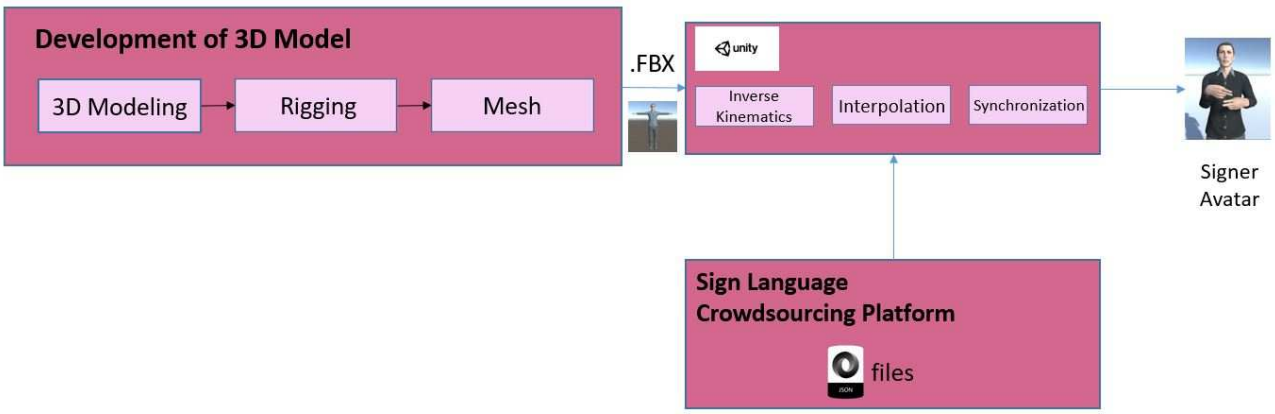

Figure 2-2 Schematic presentation for the input and output of the signer avatar service

A schematic representation for the input-output of the current architecture is given in Figure 2-2. In this figure, data are loaded from the crowdsourcing platform to the avatar, the avatar processes the data and converts them in body – hands moves and face expressions. The final result exports in an Android TV build and is adjusted as a service in EASY TV.

Regarding the type of motion data loads from the platform issues a task for the signer, the signer performs the requested signs and the capturing module records the images and extracts the motion data. As we can see there are different types of annotation procedures: the expert signer initially annotates sentences but can also break the sentences into words in a further stage. Moreover, a semi-automatic annotation mechanism can be used for making the process easier for the signer by automatically recommending the right translation for a given sign. In the end of the procedure, the data is uploaded to the sign language crowdsourcing platform and stored into repositories. The crowsourcing platform can generate additional tasks to the same signer and the whole process is then repeated.

2.2. The phases of development of a sign language avatar

The development of a sign language avatar consists of a number of phases which form a pipeline architecture where the output of each phase is given as input to the next one. The architecture includes phases for the 3D modeling of the avatar, the rig which actually is a representation of the bones, the procedure of inverse kinematics, the implementation of the motion data files in JSON format, the improvements in avatar's motion and the final build and implementation in the Android TV. A graphical representation of this interconnection between the phases of the avatar development is given in Figure 2-3

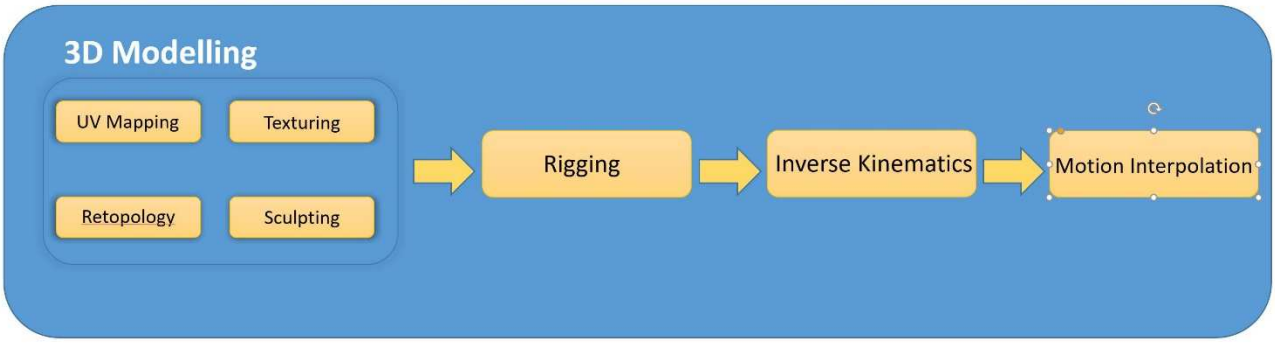

Figure 2-3: Pipeline diagram demonstrating the 4 phases of procedure for creating a 3D Avatar

More specifically, a brief description of each phase is given below:

- **3D Modelling:** First of all, a 3D humanoid model has to be created in one of the specialised 3D tools such as Blender. Regarding this project, some basic guidelines taken to be more suitable as a signer. i.e. a black shirt or gathered hair for female signer. The construction of an avatar includes three basic methods a) UV mapping[1] b) texturing c) retopology d) sculpting. UV mapping is the 3D modelling process of projecting a 2D image to a 3D model's surface for texture mapping. The letters "U" and "V" denote the axes of the 2D texture because "X", "Y" and "Z" are already used to denote the axes of the 3D object in model space. Texturing is about the surface of the avatar and how it looks, including colour and resolution. Textures are applied, to give the perception of fabric or skin, or procedurally generated materials can be used in the editors provided by the 3D tool. The next step is retopology, the method of recreating an existing mesh with more optimal geometry for the required use, for example a background not animated mesh has to have a very light mesh, with as few vertices as possible, while a humanoid mesh requires extra vertices in the joint areas and especially on the heavily animated areas such the face even if they don't add to extra skin geometry. Moreover, retopology is used to achieve optimal lighting path for a mesh. Sculpting is used to add the final details to a mesh, like brushing textures or directly sculpting a heavy mesh to alter the geometry. 3D tools offer a compact sculpting solution in order to replace the common 3D modelling procedure, which is especially useful for organic models.
- **Rig:** In order to develop a humanoid avatar, one of the most important procedures is the rigging. Rigging is the procedure of creating a skeleton consisting of a bone hierarchy. Each bone is usually parented to a previous bone which will be influencing its pose, and consists of a peak, a joint and target spatial point. Once created, the appropriate vertex groups and vertex weights for each group, usually via a method called weight painting, are assigned to the bone in order for it to have control proportionated to the control a bone or a lever would have on the same object. Mechanical rigs, for example, are designed with bones, that have full influence over their parts, as long as they are not composed of elastic materials, on the other hand organic rigs are much more elastic and the influence of each bone varies. An end-effector is the free end of the chain of alternating joints and bones and it is not a joint and an avatar can have more than one. Moreover, body parts which stabilize the rigs posture, such are the feet on humanoid avatars, are implemented as IK bones, that do not match the actual body structure, but provide a better control over the avatar. The rigging allows a 3D model to be animated in an articulated manner. An articulation is a rotation/translation of a joint which moves a connected bone. On the other hand, the pose is a set of joint articulations which results in positioning the articulated body. In the case of an avatar the rigging is the skeleton that ties in to the human posture.
- **Inverse Kinematics (IK):** IK is a method that applies in a mapped bone avatar. The Forward Kinematics is a method takes a pose as the input and calculates the position of the end effector as the output. IK is the opposite. With IK the input is defined by the target position and the IK algorithm calculates the whole pose. In current project the positions of body, hands and facial expressions defined by JSON files which include the motion data of the human signer.
- **Motion Interpolation**[2]**:** Although the positions are defined in specific timestamps and both, start and final, positions are know, in most cases some improvements have to be made so as not to have a jittery looking movement because physics and graphics are not completely synchronised. Interpolation allows to smooth out the effect of running fixes at a fixed frame rate. Physics is running at discrete timesteps, while graphics is renderered at variable frame rates. The recorded data is about 5-6 fps. So adjustments needs to be made in order to be a realistic movement.

2.3. Platform Dependencies

Unity is a cross-platform development environment featuring fundamental 2D and 3D capabilities. It was initially intended to deploy games and the most competitive feature is that its engine is extended to support 27 platforms, including Android and Web GL. In this project it will be the main program for editing the avatar, connecting with crowdsourcing platform and translating JSON files.

Blender is a 3D modeling, animation, motion graphic and rendering application. It is capable of procedural and polygonal/subd modeling, animating, lighting, texturing, rendering, and common features found in 3D modelling applications. One of the main feature has been chosen for is the capability to add and edit the rig of a 3D model.

Adobe Fuse CC[3] is a 3D computer graphics software for 3D characters development and all content available within Fuse is royalty free. The basic advantage is that the user can choose and modify character components, such as body parts or clothes, in real-time. Mixamo is a related software and it comes in after the development of the character. It is used due to its model rigging service and the facial animations library it comes with and that is the main reason it has been chosen at least at this preliminary stage to proceed to rapid development in order to focus in editing avatar inside the Unity considering the data integration.

3. 3D MODELLING PIPELINE

This Chapter provides a brief overview of the existing methods for create and edit a 3D Model. Also, a short description for Unity will be included highlighting its main advantages which led to its choice.

3.1. Creating a basic avatar

As mentioned before Adobe Fuse CC is the software which has been chosen to design a character. Regarding the Project, two characters have to be created, one male and one female, in order to take a step further to user interface personalisation. The character components can be modified in a deep and detailed way i.e. the user can change hair, color eyes even height and mouth shape. Since sign language is a particular kind of interaction some main guidelines must be observed in order to maintain a high level standard of interpretation. The main aim is that the attention of the user must not be distracted and have an excellent visual contact of hand gestures and facial expressions. So the character must not have any accessories in his hands or head and his hands and face must have clear visibility. A final customization is required regarding the texturing of the character. Figure 3 1 describes the main steps.

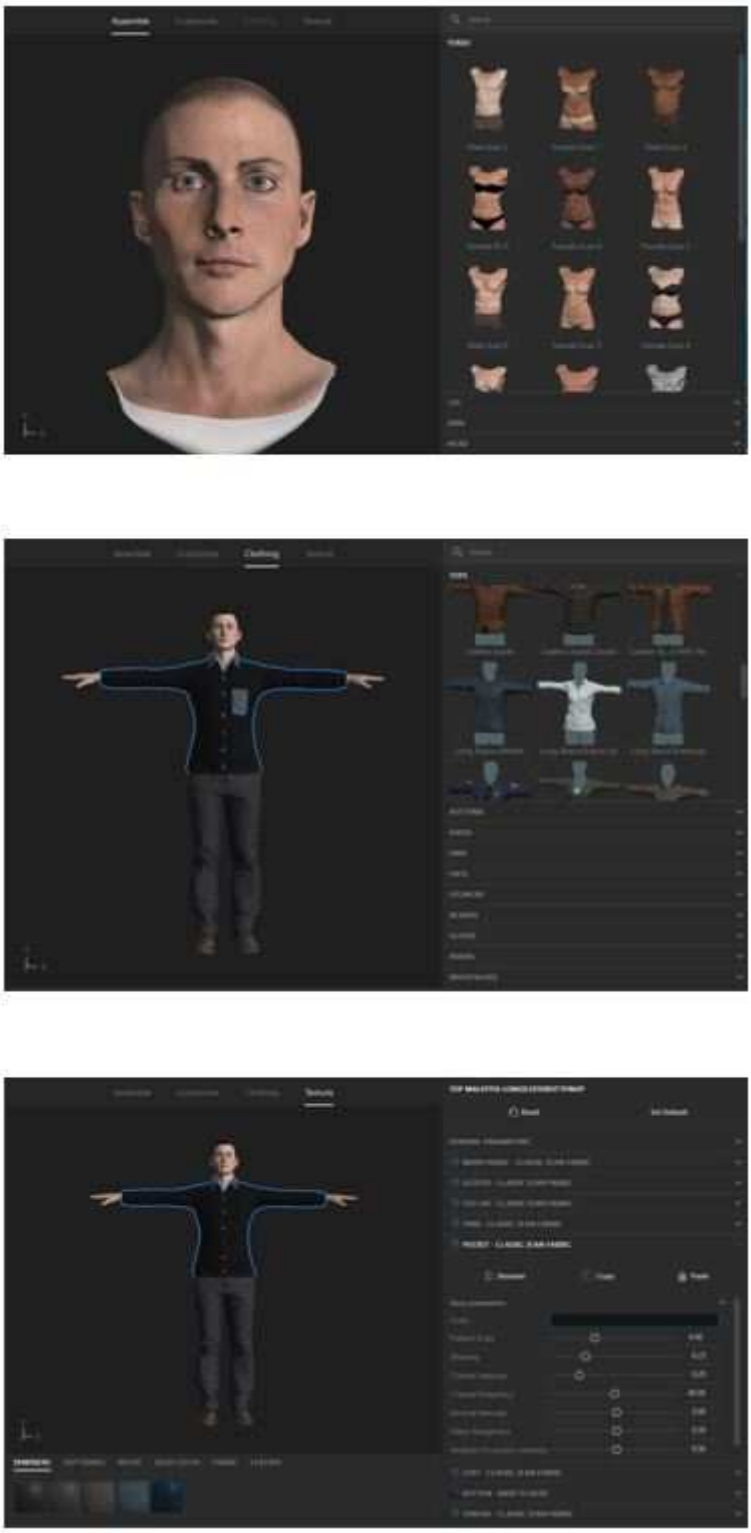

Figure 3-1: Development of a new carácter on Adobe Fuse CC

When the design is completed, the character is imported to Mixamo, another 3D software engine related to Fuse, so as to use its Rigging service. Mixamo's technologies use machine learning methods to automate the steps of the character animation process, including 3D modeling to rigging and 3D animation. The AutoRigger service applies machine learning to understand where the limbs of a 3D model are and to insert a rig into the 3D model. Another advantage of the software is that offers basic facial animation with blend shapes, which will be used to do lip synchronization. Figure 3-2 shows how a rig, with a total of 65 joints for the entire skeleton, fits to the character's body.

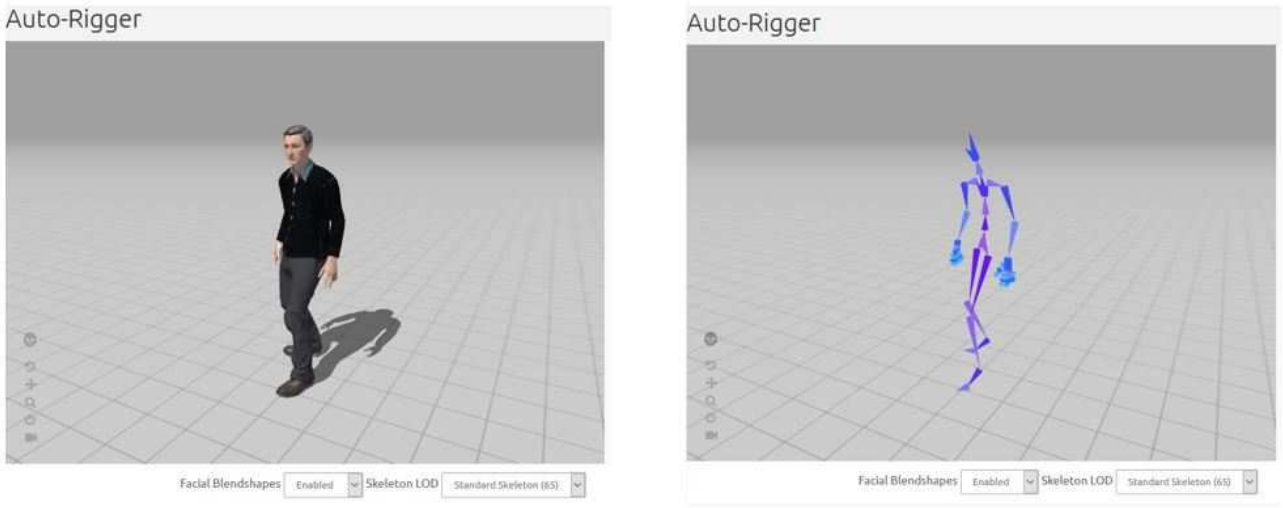

Figure 3-2: Rigging service

Another optional advantage of using Mixamo software is that offers free animations for 3D models, like jumping or running, giving a more human alike aspect. In this Project avatar's movement are actually predefined and limited to recorded signs and facial expressions as described in D3.1. "Sign language capturing technology preliminary version". Though, in order to maintain to a high standards of realism, an idle standing position will be imported to the avatar in order to use it in stand by time. The final result is a 3D rigged model which will be exported as an object for further editing or direct import into Unity.

3.2. Unity

Unity Editor is the graphic environment where the whole procedure, regarding the humanoid avatar, will take place. One of the main advantages is that Unity provides a great state machine animation system called Mecanim. Especially for humanoid animations can be applied to all models with a humanoid (human like) rig. This means that animation can easily be remapped from one character to another. It also allows for Inverse Kinematics via code as well as easy masking of limbs. Motion interpolation is also easy to be implemented through scripting offering a smoother and realistic movement. Furthermore with Unity, one can develop realistic 3D avatars capable of importing JSON files for motion playback. Lastly, Unity has network capabilities which will be very useful for connecting with crowdsourcing platform's data repository.

4. AVATAR BODY ANIMATION RIGGING METHODOLOGY

This Chapter discusses the methodology used to rig a 3D model.

4.1. Avatar Rig Construction

First of all, there are three stages for preparing an avatar in order to take full advantage of Unity's humanoid animation system. The main goal is to develop a rigged and skinned humanoid type mesh.

The first stage is modelling which is the process of creating a humanoid Mesh in a 3D modelling software such as Cinema4D or Blender. 3D Meshes are the main graphics primitive of Unity which supports triangulated or Quadrangulated polygon meshes. Nurbs, Nurms, Subdiv surfaces must be converted to polygons. To ensure that the designed model works well with animation a few guidelines must be followed as the scale of the mesh.

The second and most important stage, regarding the project, is the rigging. As mentioned before in Chapter 2.2 rigging is the process of creating a skeleton of joints to control the movements of the Model. 3D modeling software of the market provide many ways to create joints for the humanoid rig.

Some guidelines have to be followed, such as keeping the bone count to the the minimum number of fifteen bones. This is a prerequisite for Unity to be able to import correctly the rig and integrate it with its Mecanim system, by assigning an avatar to it. Inside the Unity's Editor there is the Avatar Mapping tab as shown in Figure 4-1.

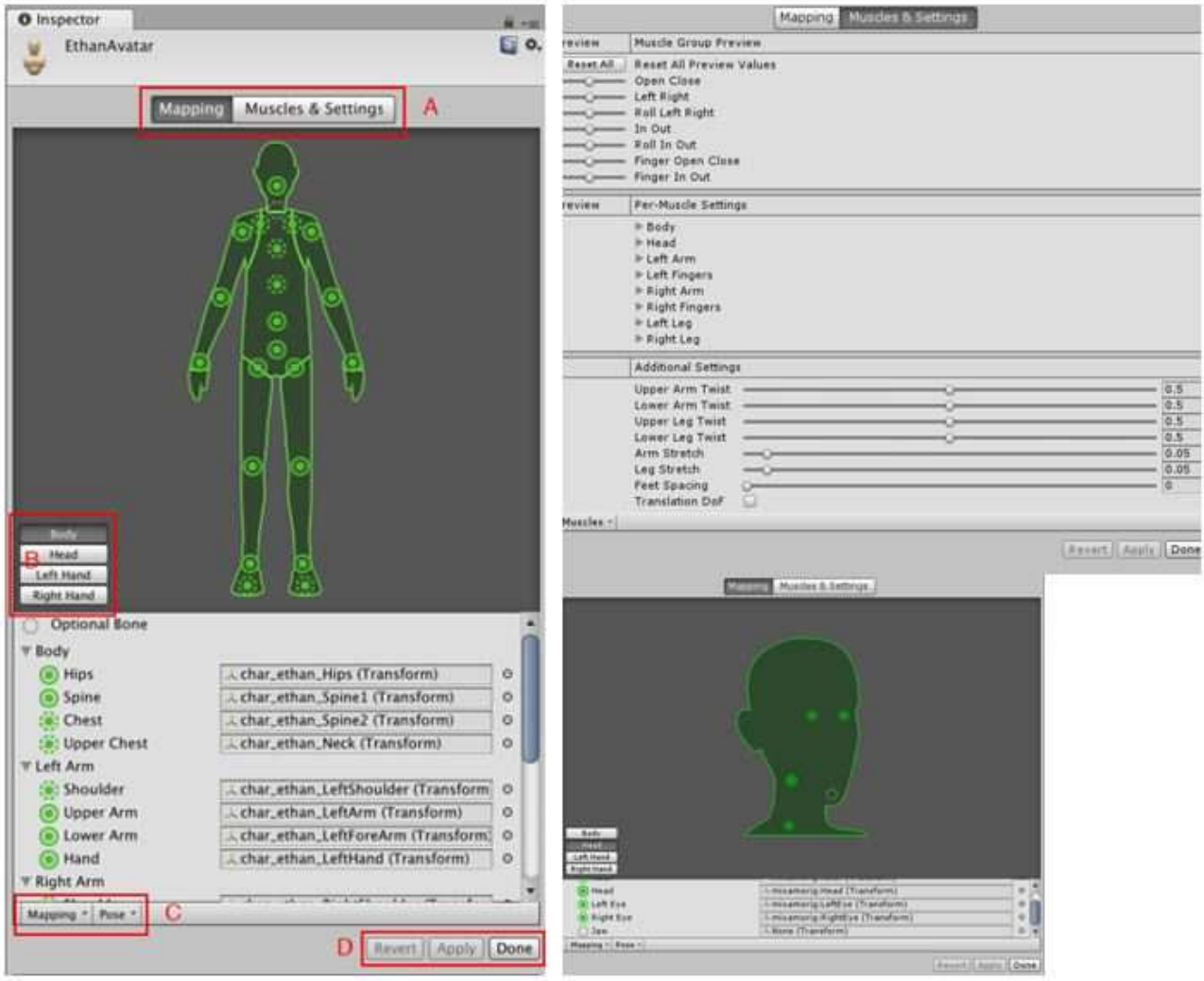

Figure 4-1: Checking the integration of the Rig

The Avatar Mapping indicates which of the bones are required (solid circles) and which are optional (dotted circles), while a more detailed image is provided for the head and the limbs, with optional bones. Another important advantage of this system is that the specific mapping can be saved and reused by another similar avatar.

Moreover, Unity provides some muscle settings which directly affect the weight of each bone and are named in a very reasonable way, instead of their anatomic properties.A rig for an impaired character could be adjusted in by tweaking some of these values. Additionally, some basic editing of the rig is provided ,which will result in direct results inside the scene, such as a character with broader shoulders.

Regarding the project two avatars will be made with the exact same avatar and rigging. Except for the optional bones a detailed strict naming convention is used for the bones in the hierarchy in order to reflect the body parts they represent and control. Unity and most 3D tools follow those conventions and ignoring them will result to subpar integration or even not being eligible for some of the animations services/plugins. Furthermore, the joint/bone hierarchy should follow a natural structure for the character entailing a consistent convention for naming the "paired" bones for example

"arm_left" and "arm_right". Expanding the definition of the articulated body and regarding its hierarchy as given in Chapter 2.2 it should be mentioned that a root joint is needed as the base of the structure. For humanoid avatars the center of the hips is defined usually as the root joint.

Regarding the Task 3.1 data positions will be recorded representing the bones and joints of an articulated model. Figure 4-2 shows the skeleton with its keypoints used to capture the recorded signs.

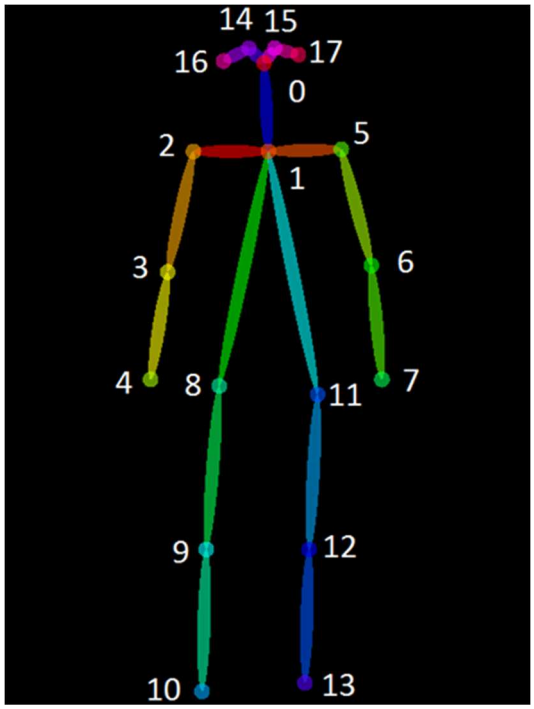

Figure 4-2: Body keypoints

Concerning the arms of the signer, keypoints for the wrists, the elbows and the shoulders are detected. Keypoints for the hips, the knees and the feet are detected for legs. Also, keypoints for the neck and the head of the signer are detected. Comparing Figure 3-2 with Figure 4-2 many similarities can be found between the two rigs. The main difference seems to be the spine bones which are not represented at all in the rig of T3.1 .

The final stage of modeling is the process of attaching the Mesh to the skeleton and it is called skinning. The main "work" of skinning is binding vertices in Mesh to the bones and that so it will have an impact to the blend shapes. Skinning usually requires a fair amount of work and testing.

Once all the 3D designing part is completed, from 3DModeling to importing the appropriate animations and testing them, there will be a fully integrated character in the project. Regarding the import procedure, Unity can import from a vast variety of different commonly used and native 3D file formats. FBX is the most common and recommended format due to advantages such as exporting animations without entangling it onto the Mesh.

4.2. Inverse Kinematics

One of the main objectives of this Project is the development of a realistic and accurate sign avatar with smooth movement so that is recognisable from a deaf user. In order to achieve the best result the Inverse Kinematics (IK) methodology must first be fully understandable. An articulated avatar is dependent on its allowed motion. In animation, motion is when an object changes position but considering a reference. As described in Chapter 2.2, when the end effector of a chain moves to a desired position a rotational or translational transformation has been applied. This procedure is the motion which has two main issues related to:[4]

- Forward Kinematics (FK) is the problem when applying known transformations to the chain

and trying to locate the end effectors' positions.

- Inverse Kinematics (IK) is the opposite. IK is described as the problem when the desired positions are know and an appropriate joint translation needs to be configured in order to reach the target positions as smoothly and as accurate as possible.

Regarding Unity's animation software (called Mecanic) for humanoids supports IK as long as the avatar is properly configured as described above. From a technical view point to set up IK for a character needs to define some target points (usually over objects the character interacts with) the character has to reach and then set up the IK through script. For example when a character grabs a pipe in his right hand and looks at it as long as he holding it. An example is shown in Figure 4-3.

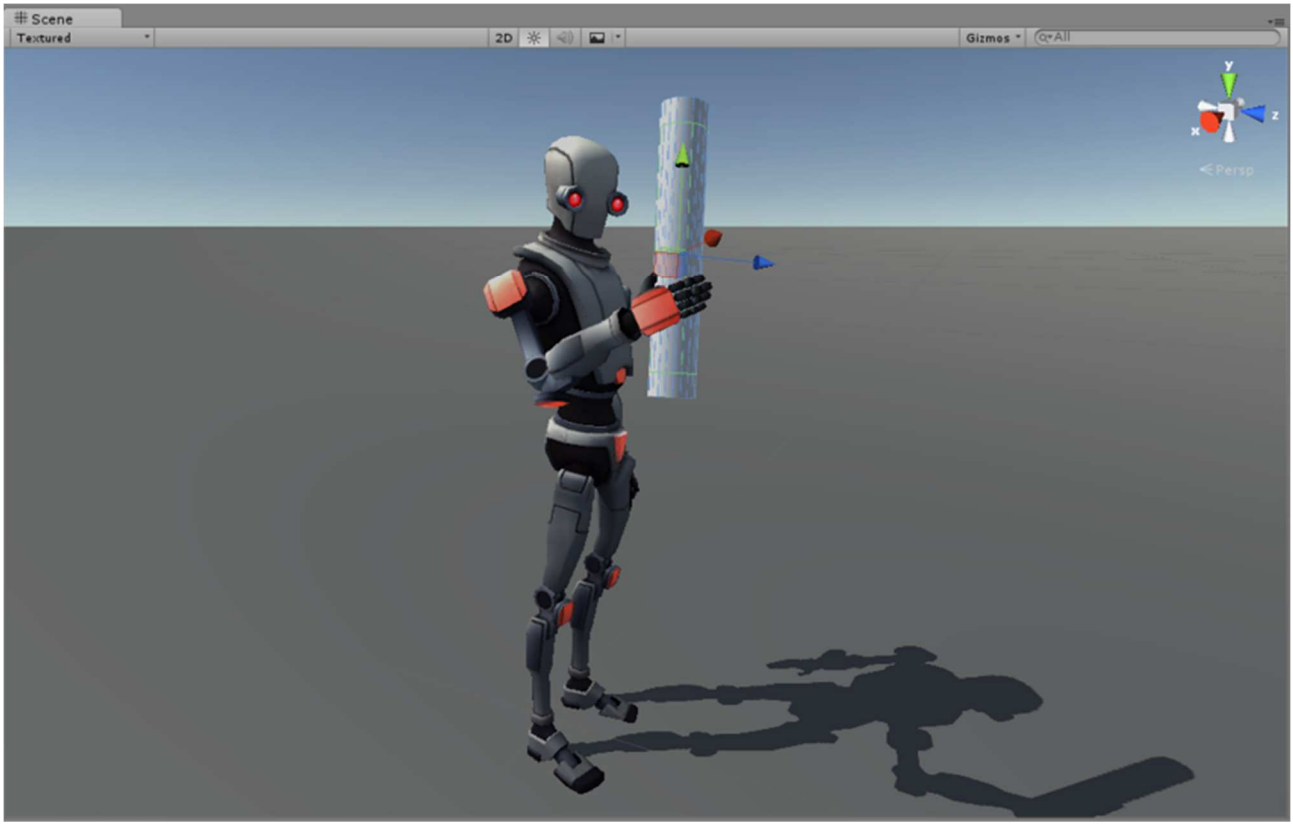

Figure 4-3: Mecanim IK grabbing

5. FACIAL EXPRESSIONS AND LIP SYNCING METHODOLOGY

Facial expressions is an integral part in sign language. Furthermore, there is a remarkable percentage of people in the deaf community who are lip readers, so the capability of lip synchronization in the module must be examined in order to increase sign comprehension.

5.1. Importance of facial expressions in sign language

Is not widely known but facial expressions plays a very important role in the meaning of a sign. The same exact hand-shape and movement can totally change meaning because of the facial expression that is used to accompany it. For example in American Sign Language, certain mouth and eye movements serve as adjectival or adverbial modifiers. In this example from Bloomberg[5] shown in Figure 5-1, signer is making the sign INCREASE, but her tight mouth and squinting eyes modify the verb to mean "increase in tiny increments." This facial expression can attach to various verbs to change their meaning to "a little bit.". In the second photo, signer is making a sign for SPILL, while at the same time making what is known as the 'th' mouth adverbial. This mouth position modifies the verb to mean "sloppily done." If you attach it to WALK, WRITE, or DRIVE, it means "walk sloppily,"

"write messily," or "drive carelessly."

Figure 5-1: Facial expressions in sign language

Despite the difference in the sign meaning, lip synchronisation is an important part of sign language. There is a number of deaf people who actually can read lips and understand the words of the speaker. In sign language trying to spell slowly each word together with the hands gesture can provide an extra help to most accurate and faster understanding.

5.2. Facial animations

One of the most important factors that make a human avatar seems real is the its ability to blend facial expressions, such as a smily or an angry face, in a realistic way. Facial animation can be achieved by presenting an array of animations, however this technique becomes repetitive and feels fake when someone is engaged in it for medium periods of time. The standard for facial animation is either a muscle blending technique like keyframe animation/BlendShapes, a bone transformation controlled animation, a motion capture system paired with a machine learning system, or system for proceduraly generated animations.[6] The difference between the array of animations and the keyframe animation is that first has predefined animations for a single model and some minimal adjustments might be needed to achieve the same result on different humanoid models, and that those animation clips have a default state from which they start and end, however some animation systems, like Unity's Mecanim, can disentangle those and access the muscles affected by the animation. The latter affects the muscles directly and exposes this feature by providing a tagged timeline of positional and rotational changes on the muscles, this has to be done in a 3D tool, where the artist defines each expression as a list of values for each muscle contraction, called a muscle macro. Some muscles may affect the same part of the skin surface, thus extra care should be taken in this procedure. Bone transformation technique require the artist to create some bones that control the muscles of the face, which in result is provided as lever to alter the facial expression. Motion Capture (MoCap) is another method which however requires the appropriate equipment. Actors are filmed to produce high fidelity facial motion, this facial data is carefully edited and keyframed by an animator. Finally, the edited data is fed to a machine learning algorithm as training data to generate our classifiers which can be used as a database of highly realistic facial animations, paired with intonation and other features that might prove usefull in the animation of the character. Once generated, the blending between two facial expressions can be achieved by interpolating the muscles of each keyframe between 2 different query results.

Unity provides Mecanim and its supporting system in order to assist the user with the integration of animations. Mecanim provides a layered finite state machine as support for blend trees, shown in Figure 5-2, which represents a simplified way of BlendShaping using Animation Clips. Given a set of parameters, a set of animations and a 2D mapping between the animations and the parameters. Depending on the type of the mapping a multivariate spatial interpolation takes place in order to blend the keyframes of those animations. Different layers of the mecanim state machine can be blended or synchronized in the IK passes (as shown by the icon next to each layer) or even set to override the above layer by providing an avatar layermask (as shown by the cog next to the layer)

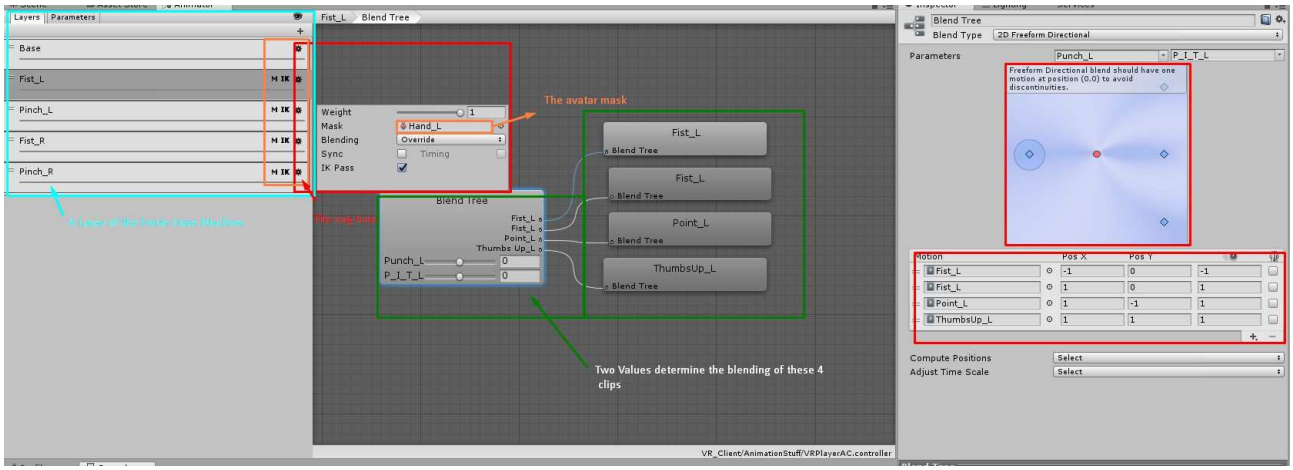

Figure 5-2: Animation blend tree

5.3. Processing of recorded data

Facial expressions are listed among the main features of the service and mainly the lip synchronisation capability. In order to remain in a high realistic level, a pretreatment is needed as far as the blended shapes of the face are concerned. In capture module the keypoints of face, described in document D3.1, are shown in Figure 5-3.

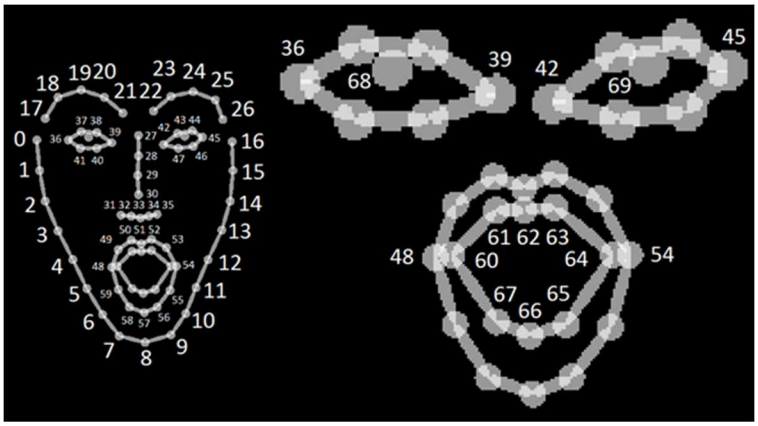

Figure 5-3: Facial keypoints

These keypoints will be stored in a JSON file inside the repository and will be the input data for the face of the avatar. The exported humanoid avatar from Mixamo has “preimported” face blending shapes, editable by Unity3D. At this stage, the recorded data regarding the facial expressions is still in progress. So a solution had to be found in order to have a first test for facial expressions of the avatar. Lip Sync Lite is one of the free assets that can be imported into Unity3D project providing better results in real-time. It is also compatible with Mixamo 3D models and, as title reveals, is specialized in lip synchronization and that’s the reason why was chosen for now. The current methodology uses is based on matching each letter of the Alphabet with the similar phonem and in its turn each of the phoneme corresponds to specific blend shapes, which is called a Viseme. For example, for phoneme “O” three blend shapes have been chosen regarding the mouth as described in Figure 5-4

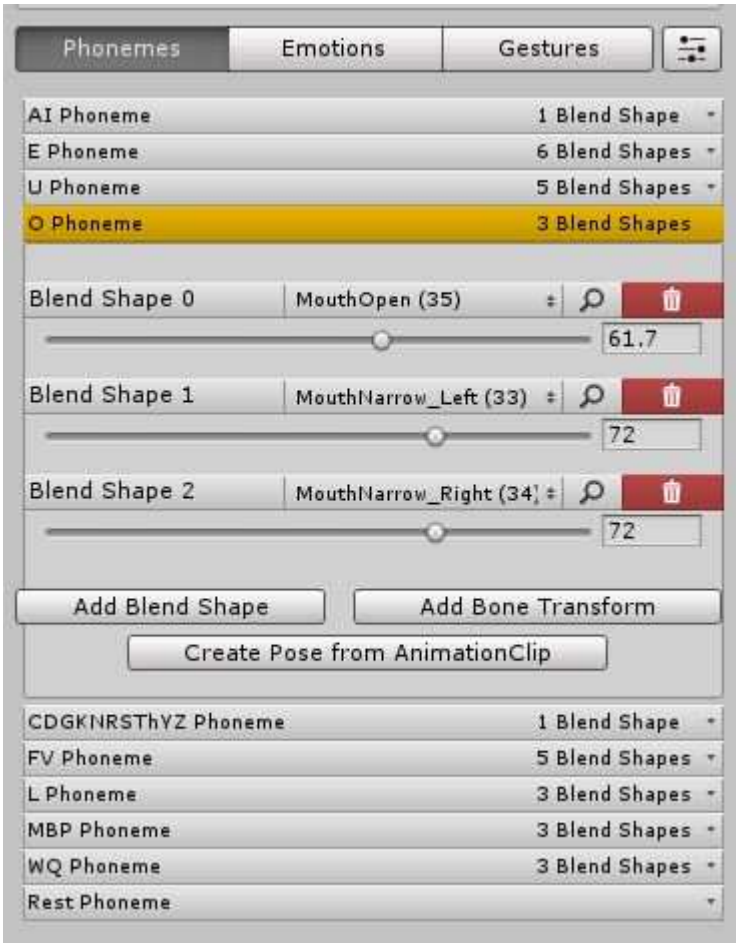

Figure 5-4: Matching phonemes with blend shapes

The choice of blend shapes was based on daily experience. In order to match each word with the corresponding phonemes, audio files were voice recorded. The content of the audio file is the translated words in each JSON file. So, each JSON file corresponds to an audio file and both of them loaded to avatar at the same time. Over the waveform of audio file, keyframes marking the position when a phoneme and its expression must be enabled matching the corresponding sound as seen in Figure 5-5

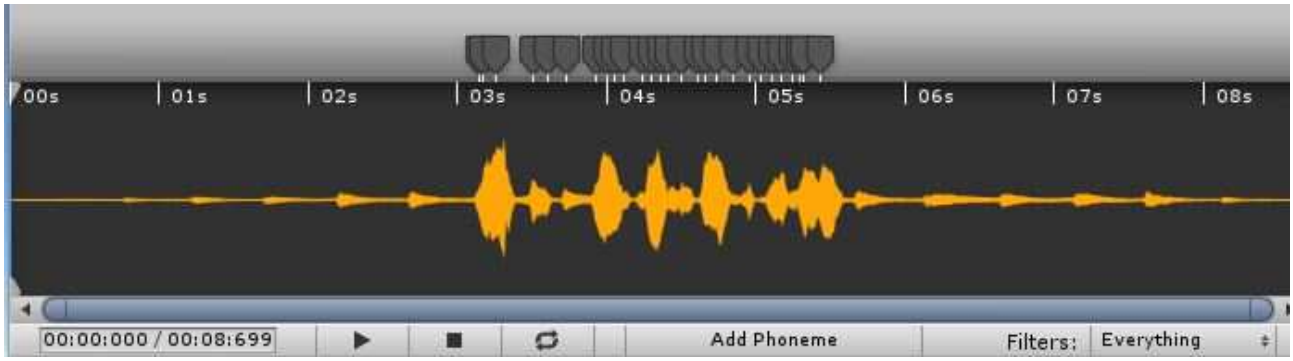

Figure 5-5: Adding phonemes in the sound file

As a final result the avatar uses continuous mouth expressions simulating the human speech. In Figure 5-6 avatar captured during “EL” in “welcome” and “T” in “to”.

Figure 5-6: Facial animations example

Besides mouth, several facial keypoints will be added, matching these on Figure 5-3 and further effort will be required to match these keypoints with corresponding face muscles. The JSON file which will be created through face recording as described in D3.1 *"Sign language capturing technology"* will contain the target positions in specific timestamps. A synchronization must be applied in order to achieve the accurate reproduction of the keypoints as they are recorded.

5.4. Motion Interpolation

The lip synchronization system providing the facial animations to the avatar is fueled by JSON files, containing the final position and rotation of the tracked limbs. The transition from the one pose to the other is done by using inverse kinematics to produce the poses in between them. The IK system once parametarized is able to smoothly bridge poses in a natural manner with regards to human anatomy limitations, like joint rotations, by solving the system of mechanical equations, defined by the constraints of the rig's joints and the added constraints to match our desired anatomy. The smooth motion is achieved via interpolation between two given tracked positions in a carefully selected timestep, while the IK system takes care of the pose by interpolating the positions and rotation of the whole joint hierarchy. A balance between the IK solver iterations and our desired visual results is decided with regards to the computationall power of the machine, which will run the application.

Interpolation is a method of curve fitting, where given an initial and a final value, an infinite amount of in between values are constructed. Interpolation methods can vary depending on the method used for interpolating (linear, spline, polynomial interpolation), the number of their variables (linear for one spatial dimension, bilinear for two, trilinear for three spatial dimensions etc) as seen in Figure 5-7 below.

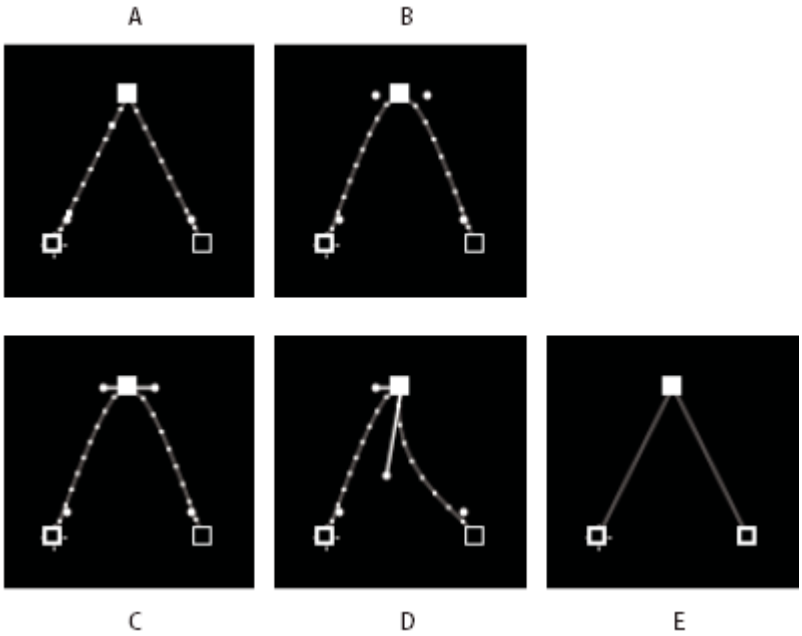

Figure 5-7:[7] **Motion path interpolation A. Linear B. Auto Bezier C. Continuous Bezier D. Bezier E. Hold**

Interpolation is very common in computer animation, usually in the form of piecewise polynomial interpolation, and is used to as described above to fill the in between frames of 2 keyframes.

6. HAND GESTURES AND SIGNS

The procedure was chosen in reproduction of recorded content for facial animations will be followed in the case of hands and body.

6.1. Processing of recorded data

As described in D3.1 *"Sign language capturing technology"* the recorded signs of the signer are stored in a JSON file. This file is stored in a repository of the crowdsourcing platform. Although at this stage a local file with recordings of hand and body joints has been used in order to test it and and create a realistic humanoid avatar. The connection of the platform will be future work.

At first, the structure of the JSON file must be analysed. An example is presented in Figure 6-1

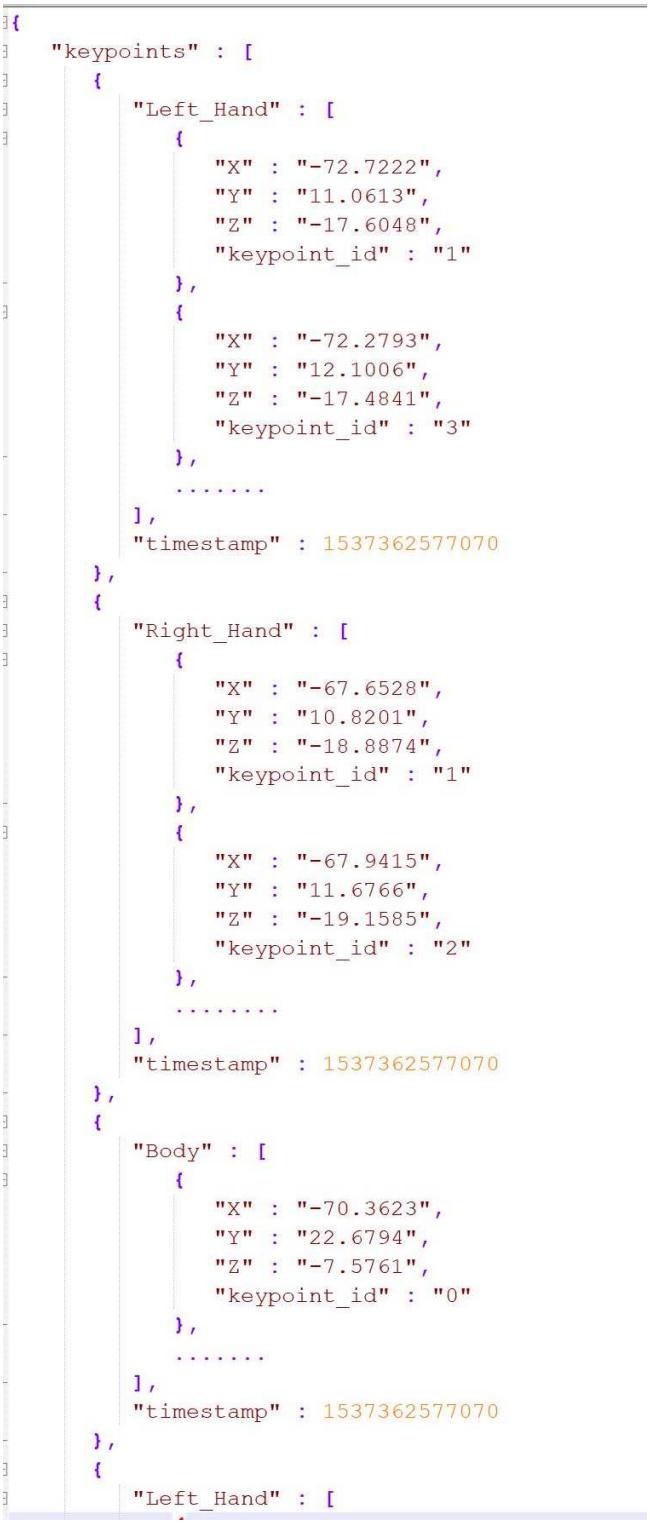

Figure 6-1: JSON file structure

There are actually three group points for the same timestamp, “right hand”, “left hand” and “body”. Each group represents the position of the specific body parts in a frame, defined by “timestamp” value. “Keypoint_id” represents the numbers as they seen in Figure 6-2 provided from T3.1.

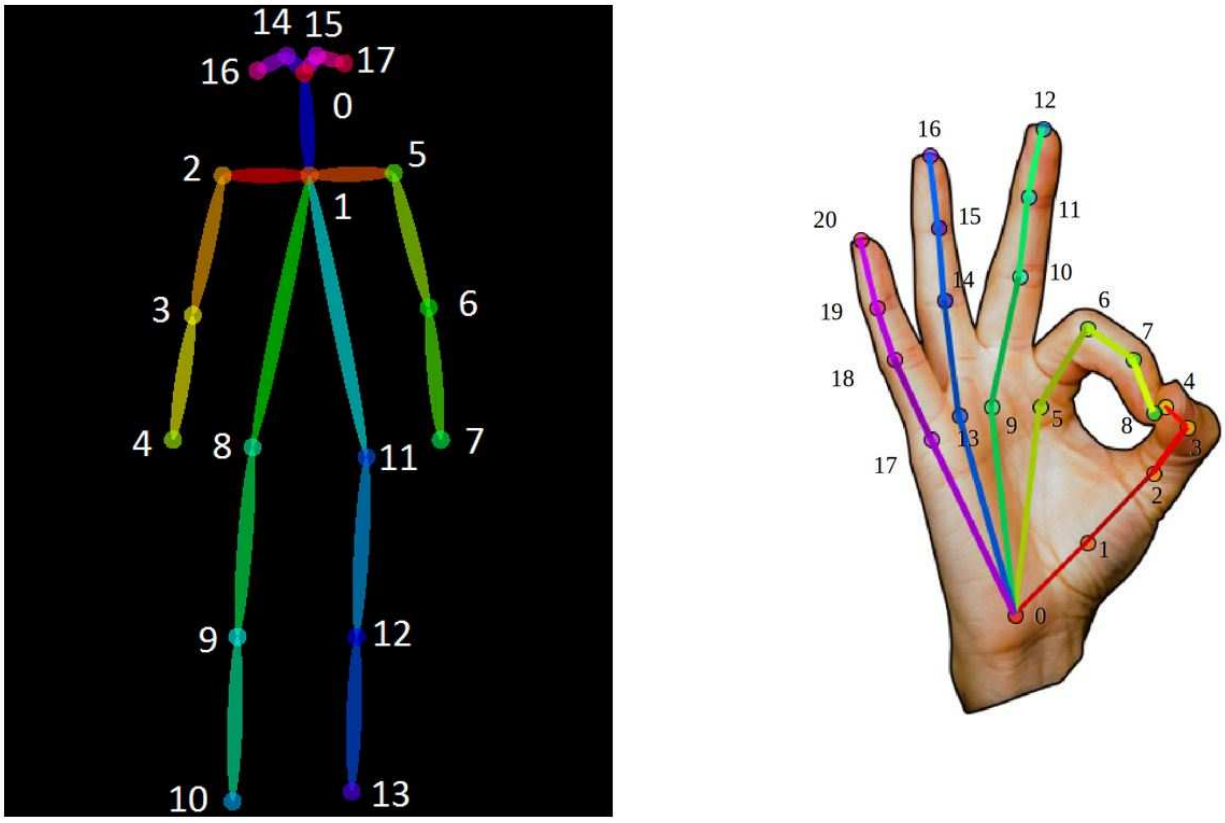

Figure 6-2: Body and hand keypoints

Keypoints are actually the joints of the avatar and the matching was done through scripting. Though, there is not a full match between the skeleton of the avatar and the skeleton of the recordings. The main difference is that the basic root joint on avatar is on the center of the hips and in recordings is on the neck. So the spine actually is never recorded. But considering that the signer always remain still there will not be any big change regarding the position of the spine. Furthermore, keypoint “1” is depicted over the rig so the majority of upper body movements reproduced quite enough. In future work, the rig of the new avatar will be accurate 100% with that in Figure 6-2

One of the major concerns was how fast the data must be loaded and reproduced by avatar. Considering that the position data recorded in a 5-6 frames per second (fps) speed, a synchronization took place in order to have the avatar human like moves and match each word to the specific corresponding sign. When the signer avatar reproduce a word from the records it is not only needed to show the whole word but also to be in a normal speed, neither too slow or fast, so as to be fully understandable by the user. Therefore it was used, through scripting, time and deltatime values in order to achieve the desirable synchronization. deltaTime is the time in seconds it took to complete the last frame. As a final result, the reproduction of JSON files regarding the time were accurate.

6.2. Inverse Kinematics

As described in Chapter 2.2 IK is used when the target position is defined and the pose of the body calculated by IK algorithm. There probably be more than one poses that the articulated body can reach the target position. Apart from time synchronization one of the main concerns in development was the use of space in order to keep avatar’s gestures in high realistic level. The recorded position points are defined in a specific 3D space. As seen in Figure 6-1 the data consists of position points in different frames which are not completely efficient in identyfing the corresponding points in Unity World environment. The difference between a subsequent position points will provide us a delta

position in world space which can be used to synchronize the positions of the avatar and the tracked marker. However, the rotation of the hands is not provided and anatomically wrong positions will result in weird final poses, thus rotational data or a the change of axis is required to achieve a visually pleasing humane pose. The body dimensions of the signer have to be taken into consideration. Humans have different body types, which furthermore change with age and gender and can result to vastly different pose approximations if the context of the body is unknown. For example a very tall human will record different keypoints, in the same word, with a normal one. Due to the above further actions should be taken regarding the way the IK algorithm operates in order to avoid non-humanoid and abnormal movement as seen in Figure 6-3 when the arm between shoulder and elbow "penetrates" into the body and the fingers joined together.

Figure 6-3: Arm penetrates the body

The first step was trying to match position points of the data with actual positions inside Unity Editor while maintaining a human pose, through scripting. Though, considering the non exact match between data position points and avatar's rigging a system for pose fitting is required, that approximates a possible pose given the motion path as defined by the 2 different tracked positions. Therefore, in many cases the result was non realistic in human like aspect. Some limitations added regarding the movement and the interactions hand and body can have but only managed to reduce the wrong gestures in some cases. The most visible important issue tried to resolve is the penetration between different body parts. An example for the actions were taken is the limitation of angles given to shoulder, in 3D space, in order to keep the arm as possible as it gets out of chest. However, the motion to target points prevailed the correct bone's position and the improved, but not completely desirable, result shown in Figure 6-4.

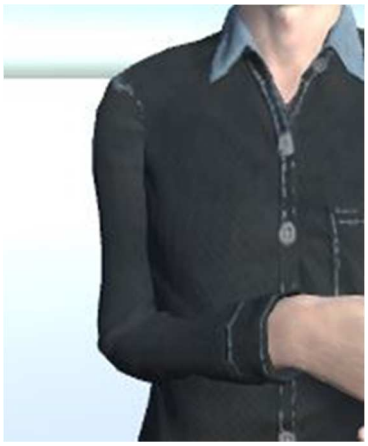

Figure 6-4 : Improved hand position

6.3. Motion Interpolation and synchronization

Contrariwise with face expressions, hand gestures and body movement are even more distinct from a fair distance in jittering picture due to their size. Also, extra care was given in cases where the movement was too large or repeating in similar position. For example the Figure 6-5 shows sign of the word “play” which is a repeated “Y” handshapes.[8]

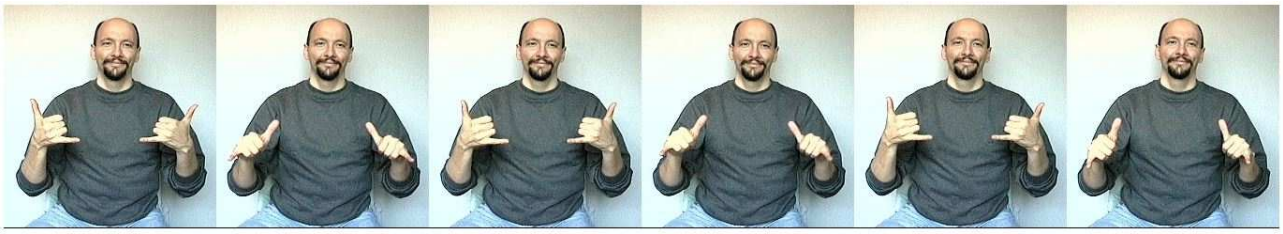

Figure 6-5 : The Word “Play” in ASL

In Chapter 5.4 motion interpolation defined and the same methodology was applied here. The start and end position each time are defined through the transition between the different timestamps. So in every transition the targeted point alternate between finish and start positions.

Lastly, a synchronization occurred between the hand gestures from the JSON file and the lip synchronization procedure. In future steps, the face expressions will be recorded and added to the same JSON file and the timestamp identification should be enough to achieve the synchronization.

7. CROWDSOURCING PLATFORM

In this Chapter we describe how the connection between avatar and crowdsourcing platform will be established through Unity.

At this stage of development the implementation of JSON files took place only locally. As described in Chapter 2, data repository of crowdsourcing platform will be the input for the avatar module. A user who uses the service will have the option to choose the sign language he prefers. Therefore, the avatar service must be capable of connecting through network with data repository and receives all the updates. Precautions must be taken regarding the connectivity and how the service can handle connectivity problems such as low speed connection. The synchronization between hand gestures and facial expressions must be continuous otherwise the final result will not be understandable from the user. Unity can handle network services and JSON files as mentioned above.

Another issue will be the multi-language capability of the platform. Data files must be stored properly and with specific identification in order for the avatar service to distinguish the content the user asks. Therefore, if a user uploads some recordings using the capture module, options must be given so as to categorize it properly. The service of the avatar in turn must be capable of choosing the corresponding files that was asked for.

A connection with the personalized access services of T5.2 would probably offer further options regarding the automation of the service. Expanding the role of data repository could also offer many advantages to the avatar. For example, the data from T3.1 could be populated with text representing each recorded sign word. T3.2 in its turn could translate the words to each language. As a result a text subtitle could accompany the avatar during the avatar. Having text inside repository could be also automated the lip synchronisation methodology. Furthermore, a text-to-speech tool, similar to T2.3 Screen reader tool, could be added offering audio in the signer avatar.

8. CONCLUSIONS AND FUTURE WORK

Concluding the project at this stage, a humanoid avatar is created and his movements submitted by a local JSON files which includes specific recorded movements representing words in sign language. Furthermore, facial expressions added and synchronized with hand gestures, aiming at the lip synchronization. Figure 8-1 Shows an avatar presenting the word “name” in sign language.

Figure 8-1: Word "name" in sign language

The next step is the development of a new avatar from scratch focusing on the accurate representation of the keypoints of the body, hands and face as described in capturing module with the corresponding joints, bones and points the rig will have. It must be discussed if at least one more keypoint can be added in the center of the hips or somewhere over the spine because it will help to get a more accurate position for the upper body. Regarding the face, it must be decided which methodology, between bones transformation or blend shape, will be applied. There will be tests with the new data locally and all the cases will be examined and the more realistic will be followed.

The most important issue for future work is the interconnection between avatar and platform's repository, not only regarding the connection itself but also the data's content. At this stage, position data actually covers just a part of the information the avatar needs to move accurately. It should be re-examined the content of a data a record can take in the capturing model level and at the same time to optimize the scripting on the avatar level regarding the IK and motion interpolation as described in the Chapter 6. Furthermore, it should be decided the use of a generic model commonly accepted between the two modules. The recorded data a user can upload must be converted to this model otherwise there will be problems regarding scale and accurate positions. About the connection with platform the technical details as the protocols will be discussed and tested considering also Unity's specifications.

Another concern is the quality of the final reproduction of the recorded data. There must be a comparison between the original captured video and the avatar's signs. Also, extra notice must be given in the word transition from repository. It should be discussed if the data positions in the first timestamp describe an idle position or the beginning of the sign. In sign language the transition from one word to another happens instantly. So there must be a second level of checking realistic and accurate word's transition of the signer avatar. Apart from avatar, a background must be chosen such as not to distract the user and to maintain a colour harmony with the environment the user sees.

The avatar must be exported in WebGL format in order to be accessible via the main terminals.

Lastly, considering the D1.3 document, extra features may be discussed coming from T5.2 like the text-to-speech capability and the necessity or not of adding audio files to come along with signs.

9. REFERENCES

[1] WikiPedia, “UV Mapping.” [Online]. Available: https://en.wikipedia.org/wiki/UV_mapping. [Accessed: 30-Sep-2018].

[2] Unity3D, “Interpolation.” [Online]. Available: https://docs.unity3d.com/ScriptReference/Rigidbody-interpolation.html. [Accessed: 30-Sep-2018].

[3] “Mixamo.” [Online]. Available: https://en.wikipedia.org/wiki/Mixamo. [Accessed: 30-Sep-2018].

[4] A. Arsitidou and J. Lasenby, “Inverse Kinematics a reviewof existing techniques and introduction of a new fast iterative solver,” *Univ. Cambridge*, 2009.

[5] A. Okrent, “Why Great Sign Language Interpreters Are So Animated.” [Online]. Available: https://www.theatlantic.com/health/archive/2012/11/why-great-sign-language-interpreters-are-so-animated/264459/.

[6] J. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, “Practice and Theory of Blendshape Facial Models,” *Eurographics*, vol. xx, pp. 199–218, 2014.

[7] Adobe, “Key Frame Interpolation.” [Online]. Available: https://helpx.adobe.com/after-effects/using/keyframe-interpolation.html.

[8] “ASL - ‘PLAY.’” [Online]. Available: http://lifeprint.com/asl101/pages-signs/p/play.htm. [Accessed: 30-Sep-2018].