Sign Language Translation Mobile Application and Open Communications Framework

Deliverable 5.2: A Virtual Character

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101017255

Project Information
Project Number: 101017255
Project Title: SignON: Sign Language Translation Mobile Application and Open Communications Framework
Funding Scheme: H2020 ICT-57-2020
Project Start Date: January 1st 2021

Deliverable Information
Title: A Virtual Character
Work Package: WP 5 - Target Message Synthesis
Lead beneficiary: UPF
Due Date: 31/12/2023
Revision Number: V1.0
Authors: Josep Blat, Víctor Ubieto, Eva Valls, Jaume Pozo, Oriol Jimenez, Carolina Diaz del Corral
Dissemination Level: Public
Deliverable Type: Demonstrator

Overview: This document is the continuation of D5.1, which discussed the details about an improved new avatar in terms of skeleton, hands, skin and eyes. In this deliverable, we focus on character customisation, and the approach followed to support diversity of avatars. In addition, it is also discussed the adaptation when integrating high quality avatars in mobile devices. Lastly, we provide the final

details left in D5.1 about the improved avatar. Specifically, we talk about hair rendering and morph targets (blendshapes).

Revision History

Version #	Implemented by	Revision Date	Description of changes
v1.0	Josep Blat	08/12/2023	Integration of changes suggested during reviewing towards deliverable submission
v0.2	Josep Blat	30/11/2023	Version for partners' review
V0.1	Víctor Ubieto	27/11/2023	Initial version

The SignON project has received funding from the European Union's Horizon 2020 Programme under Grant Agreement No. 101017255. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SignON project or the European Commission. The European Commission is not liable for any use that may be made of the information contained therein.

Approval Procedure

Version #	Deliverable Name	Approved by	Institution	Approval Date
V0.1	D5.2	Shaun O'Boyle	DCU	28/11/202x
Vx.x	D5.2	Marco Giovanelli	FINCONS	07/12/2023
Vx.x	D5.2	Vincent Vandeghinste	INT	6/12/2023
V0.1	D5.2	Adrián Núñez-Marcos	UPV/EHU	27/11/2023
V0.1	D5.2	John O'Flaherty	MAC	27/11/2023
V0.1	D5.2	Santiago Egea Gomez	UPF	05/12/202x
V0.1	D 5.2	Irene Murtagh	TU Dublin	27/11/2023
Vx.x	D5.2	Anthony Ventresque	TCD	7/12/2023
Vx.x	D5.2	Mathieu De Coster	UGent	28/11/2023
Vx.x	D5.2	Jorn Rijckaert	VGTC	01/12/2023
V0.2	D 5.2	Henk van den Heuvel Louis ten Bosch	RU	27/11/2023
Vx.x	D 5.2	Lien Soetemans	KU Leuven	29/11/2023
V0.1	D5.2	Rehana Omardeen	EUD	28/11/2023
V0.2	D5.2	Mirella De Sisto Dimitar Shterionov	TiU	05/12/2023

Table of Contents

1. Introduction

This deliverable follows D5.1 *"A Virtual Character",* submitted in M18. Our characters need to be animated to produce sign languages, and thus the perceived quality by users relates both to that of the character aspect / appearance and that of the signed animation (and their interplay). The quality needs to take into account the *uncanny valley*, as already discussed in D5.1, a topic which we do not discuss further here. Our main focus in the project has been on the naturalness of the animations of the signs, but the visual quality of the avatar/character is important, as well as the possibilities of its customisation. In D5.2 we discuss the progress in the quality of the characters, both visual and related to animation. We especially discussed the advances towards customisation, which has been an important user's requirement.

Before turning to these aspects, we revisit the user requirements in Section 2, in keeping with our user (Deaf/Hard of Hearing, DHH) driven project focus.

As in D5.1, the skeleton definition, key for high quality animations, plays a major role. Our focus in D5.2 relates to the need of supporting diverse characters, while keeping interoperability, evolving from the single character system presented in D5.1. This requires an approach that adapts animations to the different characters. A first aspect is related to the multiple types of skeletons, naming conventions and reference poses to start the animations from which are currently in use by different widely used platforms. To address this issue, we have identified and clearly documented a reference skeleton, including a reference facial representation. A second aspect relates to the importance of specific locations in multiple communicative signs (such as pointing to the heart). These locations vary for the diverse characters and this issue has to be taken into account. Our software has been re-factored to deal with these issues. A key ingredient is a configuration file associated to each character (this approach already existed in the JASigning system rendering SiGML), and the ability of the software to support these configuration files and perform the animated signs correctly. We have created tools to support the creation of this configuration file easily as well as extensively documented our approach and references, which is less extensive in the JASigning case, which also has scarce support for Non Manual Features (NMFS). This is discussed in Section 3.

Beyond technical interoperability when using multiple characters, they should exist to support customisation by and diversity of users. Support to diversity implies taking into account multiple visual characteristics associated to gender, race, age, size, … which is a challenging issue to deal with in a cheap and scalable way. Also in Section 3 we describe and discuss in detail a pipeline to generate a wide variety of high-quality characters. It is based on Open Source models, and multiple Open Source software. As it will be seen in the section, connecting these different sources and complex tools in a way that the outcome is high quality and allows for interoperability has required significant development, trials, testing, and error analyses. However, the procedures of the proposed pipeline can be very precisely described, and largely scripted, so that a customisable character can be created by an experienced operator in a relatively short time. In this way, we provide a scalable and high-quality solution to character customisation, which, as foreseen in D5.1, should be central in D5.2. This contribution will enable the co-design of avatars with users, to be carried out in the future.

Another step in making characters more practically usable lies in adapting them to mobile use - where resources are scarce. In Section 4 we discuss the pipeline aspects to turn the desktop based avatars to mobile ones.

Finally, we revert back to our initial goal, which was based on creating a new character, starting from an initial modification of an existing character, EVA. The process of developing a second, different, and much higher quality avatar in terms of animation and visual quality, has informed us in defining the interoperable skeleton and face, as well as providing insights to drive the quest of the visual quality pipeline. In Section 4, we discuss the final progress of our second character, and how it has fed into the other results discussed in this deliverable.

In summary, this deliverable provides a pipeline for creating a full range of quality customisable avatars, a strategy to make avatars interoperable within our open source software for sign synthesis, the final details of our high quality one, and the adaptation of avatars to mobile devices requirements, in the framework of the discussion of the user requirements elicited within SignON, to which we turn next.

2. User Requirements Revisited

In this section, we take stock of the user requirements with respect to the signing avatar elicited within WP1. Taking into account feedback and comments from the DHH community has been and is the highest priority within SignON and, consequently, in this report we firstly review these requirements and position our work with regards to them. The users' concerns and opinions should guide the focus on the avatar appearances to be addressed. Another aspect that has impacted significantly the development of our work has been the project needs related to machine translation (MT), the little availability of Open Source resources for signing avatars, and the scarce documentation available for those resources found. Thus, we gave priority to the work of improving the procedural animations, where existing resources could be turned into signed animations, over the visuals of the avatars. However, as we will present in the following sections, within the project we also created tools to support avatar diversity.

Our main source for user requirements are those collected in Deliverable 1.8 “Final User Requirements Report”, dated in June 2023 (we had direct contact with users in the co-creation event organised by EUD in Madrid, where we presented some of our technologies). We analyse them along two different aspects, customisation features and visual features, both to facilitate the discussion in this section, and to refer to them more compactly later within this document.

In terms of customisation, users wish to have different avatars, at least to support gender variation and multiple cultures. They also demand to be able to control signing attributes such as the signing speed. We have developed support for the first wish by providing two body types of avatars. Users can choose the avatar to use when opening the app in future versions. Additionally, signing attributes control can be interactively performed in our real time signing synthesis system, so that users could manage the signing speed or the dominant hand during execution time.

In terms of how the avatar looks visually, there are contradictory opinions. Some users are opposed to human-like avatars, while other users prefer to have realistic avatars that look as human as possible. Our position is that we should try to provide the users with high-quality avatars. Over the last couple of years, there has been a remarkable improvement in realistic characters (e.g., Unreal Engine Metahumans), and we believe that sign language projects/work should not be excluded from doing the same. Nevertheless, users and ourselves are aware that their level of quality should match the animation, so that the final result looks natural and does not give a sensation of creepiness (this was discussed more in depth in D5.1). Nonetheless, users strongly pointed out that quality of sign language production should be a priority over the looks of the avatar. Again, we agree with this statement, and the evaluation of the animations is extensively discussed in D1.6. A final requirement is that there should be some way to control the clothing colors and background to enhance the visibility of the signs. Our developments cater for this, and this can be incorporated into future versions of the app to ensure that the avatar is also accessible for DHH users with some degree of visual impairment.

Let us remark that users have the view that facial expressions (which, as other NMFs, are key to sign language grammar) are a priority over the customisation of the avatar. In more technical terms, this means that facial animation should be able to result in performing all possible face states as a priority. This aligns with our priority to the animation work, which links with the work in Section 3.

3. Avatar Customisation and Interoperability

As discussed in Section 1, our previous work within SignON has been focusing on maximising the quality of sign animations for a single avatar (Eva). This freed us from tackling multiple issues associated with using several characters initially, which were not desired. In this section, we present the pipeline developed to fully use different avatar models into our sign synthesis system, the realiser, recently renamed as *Performs*. More precisely, in order to change to a multiple character paradigm, there are two complementary tasks: obtaining a multiplicity of new 3D characters that we would like to use as the signing avatar at synthesis; and implementing a robust system that is prepared to perform correctly when the avatar changes.

3.1. Obtaining 3D avatars

By definition, the SignON app is a machine translation (MT) application which runs on mobile devices. The customisation of high-quality characters/avatars in mobile devices is neither a standard practice nor common one. The hardware limitations make it difficult to provide a satisfactory customisation system, both in terms of power and user interaction. The performance suffers when rendering high-quality avatars in a mobile environment, and the small screen does not allow the users to appreciate details that would enhance the visuals of the avatar otherwise. Furthermore, the customisation interface is a problem because it fights for space/area with the view of the avatar, which should be maintained when customising it. Moreover, large assets and memory are required to support the different customisation parts, which would contradict with this project needs, in particular, very reduced loading time.

Different ideas were explored, such as reducing the customisation of the avatar to make it fit correctly in the application, or creating a separate web-based application that connected with the app and synchronise the avatar. But these ideas did not fit well in the context of the SignON project because they contradicted the user requirements or because they would heavily impact the mobile app performance. Therefore, our current approach is to create a separate strictly standardised pipeline using Open Source resources that will generate avatars whose use in our system is supported. This way, we make sure that those avatars (and therefore animations) follow the same standard skeleton, while it requires the minimum memory to load a new avatar to use. The former is most critical, since it means that it saves us quite a bit of computing needed when retargeting animations between different avatars and it results in more reduced loading time and faster performance than manipulating or sculpting an avatar on the fly.

Retargeting is the process to map animations between different avatars. Its complexity increases if the skeletons of the different avatars are not at the same rest position, if the bones have different length, if the skeleton hierarchy is different, etc. By using our approach, which we summarise below, this problem is addressed.

The pipeline allows users to create, or use a variety of, 3D character models, properly adjusting a skeleton to the model chosen, improving the skeleton quality for using a repository of high quality animations and test its suitability for animations; adjusting finer quality aspects to be able to support sign languages and export a file (in .glb format) that can be an input to our system. The way the realiser Performs has been developed, supporting web interoperability, facilitates that the avatar change can be easily supported by the SignON mobile app. Through an interactive web, naming conventions and special positions are identified so that the avatar has a configuration file that can be imported to easily use the avatar in our sign synthesis system. Figure 1 visually represents the different processes. We give some detail next, while considerably more detailed documentation of this pipeline is available as an annex.

1) Creating, reusing or modifying high quality 3D character models. **Character Creator 4**[1] is a full character creation solution for designers to easily generate, import and customise stylised or realistic character assets for a variety of 3D purposes. When the user is satisfied, a file is exported. While the character mesh is visually appropriate, the following step of the pipeline is needed to get a much better skeleton to support the animations.

2) **Blender**[2] is a free and open-source 3D computer graphics software toolset used for multiple 3D applications. In our pipeline it is used to remove the armature/mesh from the associated skeleton, perform a much better rigging of the 3D model, or add bones which are needed.

[1] https://www.reallusion.com/character-creator/
[2] https://www.blender.org/

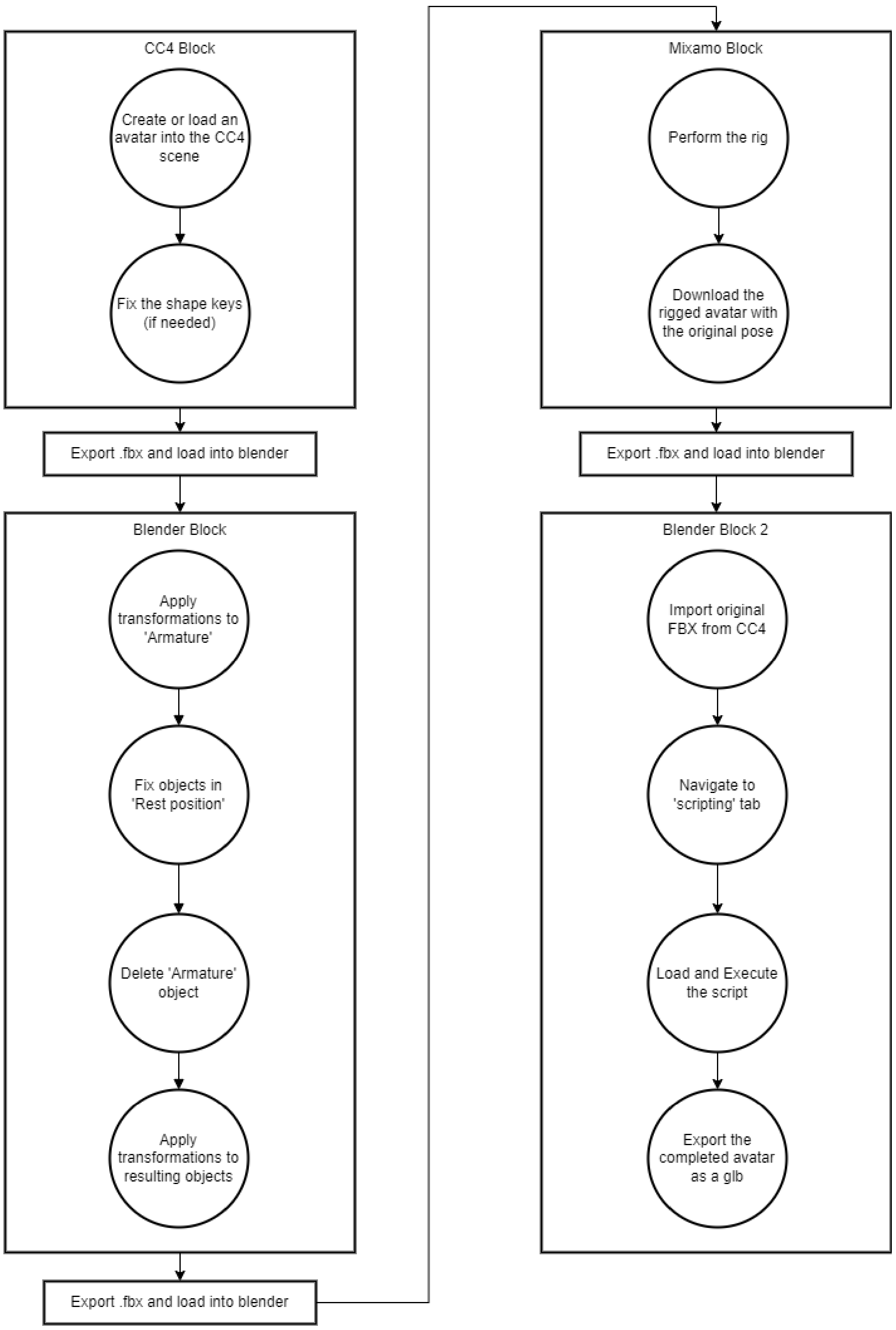

CC4 Block
Create or load an avatar into the CC4 scene
Fix the shape keys (if needed)
Export .fbx and load into blender
Blender Block
Apply transformations to 'Armature'
Fix objects in 'Rest position'
Delete 'Armature' object
Apply transformations to resulting objects
Export .fbx and load into blender
Mixamo Block
Perform the rig
Download the rigged avatar with the original pose
Export .fbx and load into blender
Blender Block 2
Import original FBX from CC4
Navigate to 'scripting' tab
Load and Execute the script
Export the completed avatar as a glb

Figure 1: Visual diagram of the pipeline to obtain multiple 3D models to use in the SignON system.

3) **Mixamo**[3] is a widely used open-source platform that offers a repository of previously made 3D animated characters and an automatic rigging system for custom avatars. Mixamo is employed to integrate a default skeleton into the avatar. By using it, we set as default a complex enough armature (a total of 67 bones: legs 10: left 5, right 5; torso + head 15: neck incl. eyes 5, left shoulder to hand 3, right shoulder to hand 3, Hips + Spine 4; hands 42: left hand incl. 4x5 fingers 21, right hand 21) that performs animations without visual errors and we ensure that our generated animations can be used by any Mixamo user and, consequently, by any Mixamo avatar, facilitating the interoperability of our system.

4) Next, **Blender** is again used to apply a custom script that corrects and standarises different aspects of the 3D character.

Now that the whole pipeline has been explored and tested, and the complementary tools developed, an experienced operator performs these steps in approximately 15 minutes for any new avatar. After these general purpose steps which lead to obtaining multiple avatars with a reference skeleton hierarchy, bind position, rig, and blendshapes, they need to be imported to our sign synthesis system, which has some additional requirements. We discuss this in the following subsection.

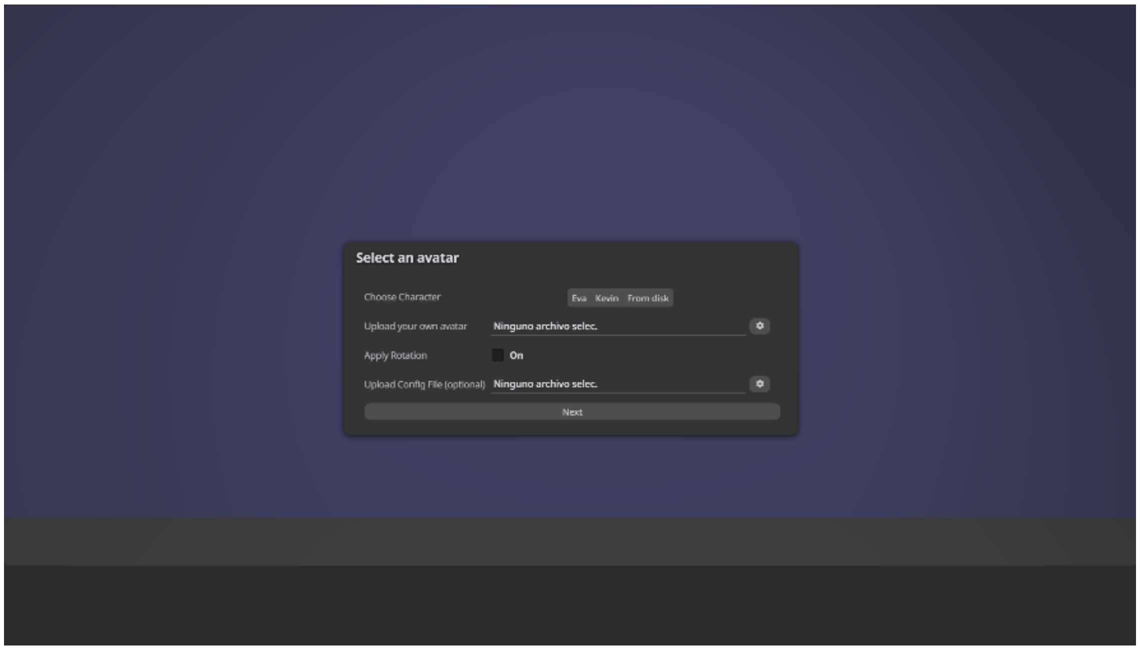
3.2. Supporting different 3D avatars in the sign synthesis system

The synthesis of the signed animations relies heavily on certain naming conventions and positions in the 3D space surrounding the avatar. Indeed, each avatar might have different names to refer to the same element, different body proportions or even different distribution of meshes, possibly making existing animations for one character worthless for another character. As a way to deal with this problem, we can supply a file for each model which acts as a bridge of configuration between the system and each avatar, both for manual and non-manual features (MFs and NMFs).

Of course, these issues are reduced to a minimum by following our proposed pipeline to obtain an avatar. This way, the support is almost direct, except for the setting of the spatial positions (which depends on the avatar proportions). Nevertheless, we maintain the possibility for importing other configurations, to facilitate the use of a 3D model coming from external sources, to widely expand the

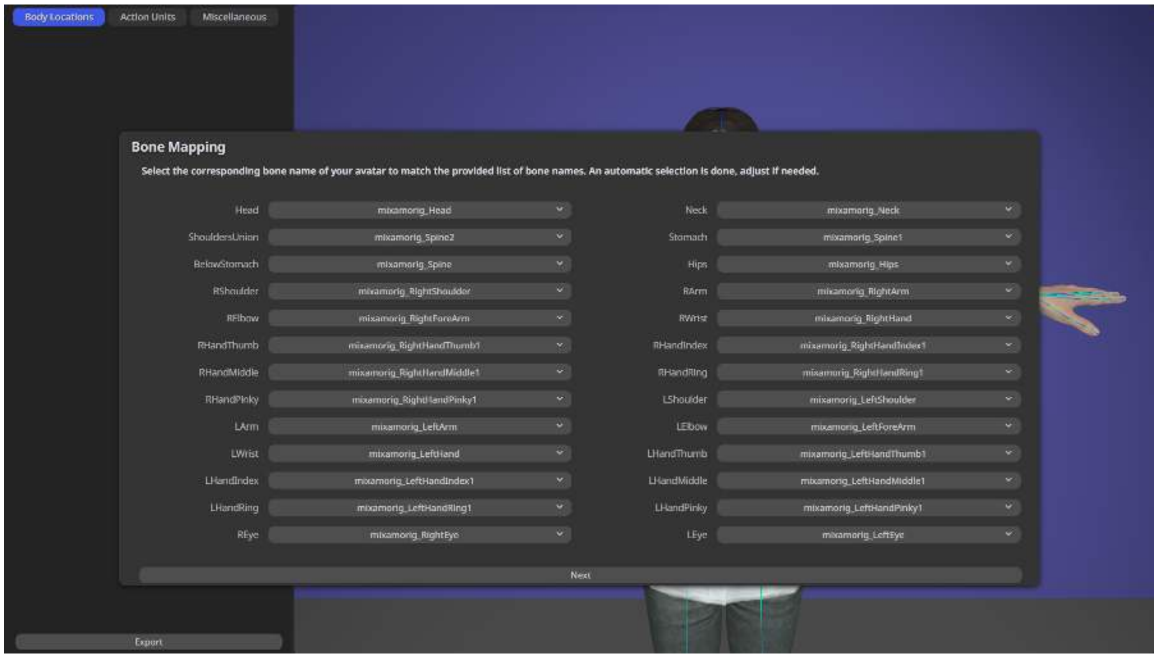
[3] https://www.mixamo.com/

range of avatars that our system could automatically support. In order to facilitate the avatar configuration by skilled users, we have developed a webpage with a practical GUI that creates the configuration file automatically based on user's inputs.


Select an avatar

Choose Character	Eva Kevin From disk
Upload your own avatar	Ninguno archivo selec.
Apply Rotation	On
Upload Config File (optional)	Ninguno archivo selec.

Figure 2: Load screen of the configuration file webpage.

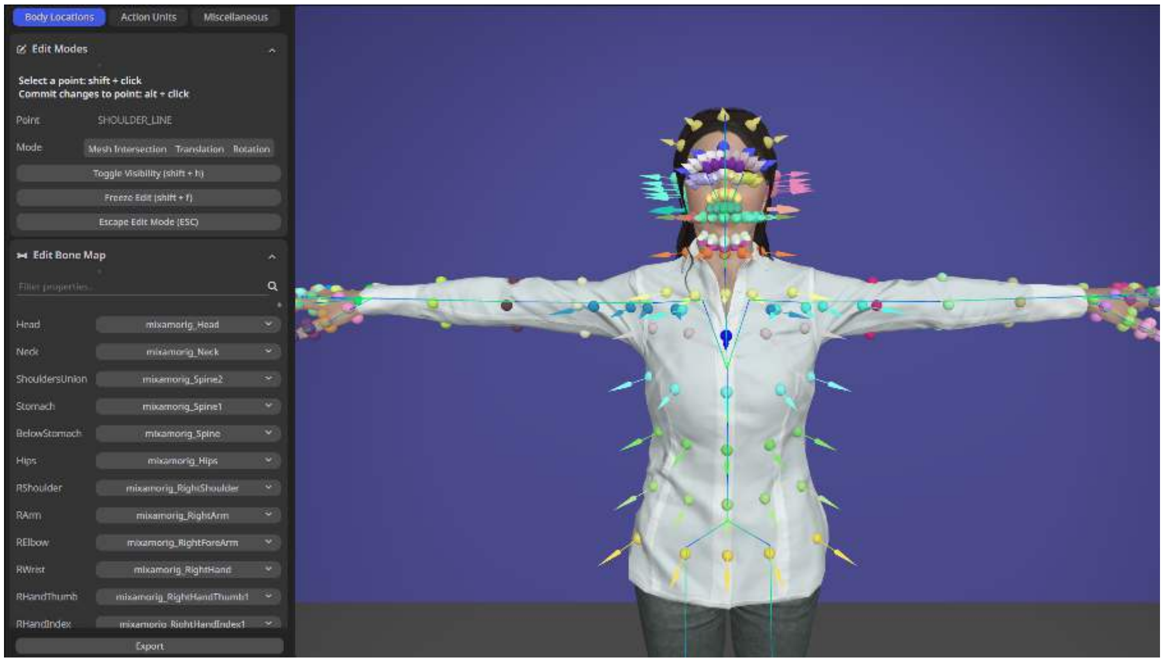


Bone Mapping

Select the corresponding bone name of your avatar to match the provided list of bone names. An automatic selection is done, adjust if needed.

Head	mixamorig_Head	Neck	mixamorig_Neck
ShouldersUnion	mixamorig_Spine2	Stomach	mixamorig_Spine1
BelowStomach	mixamorig_Spine	Hips	mixamorig_Hips
RShoulder	mixamorig_RightShoulder	RArm	mixamorig_RightArm
RElbow	mixamorig_RightForeArm	RWrist	mixamorig_RightHand
RHandThumb	mixamorig_RightHandThumb1	RHandIndex	mixamorig_RightHandIndex1
RHandMiddle	mixamorig_RightHandMiddle1	RHandRing	mixamorig_RightHandRing1
RHandPinky	mixamorig_RightHandPinky1	LShoulder	mixamorig_LeftShoulder
LArm	mixamorig_LeftArm	LElbow	mixamorig_LeftForeArm
LWrist	mixamorig_LeftHand	LHandThumb	mixamorig_LeftHandThumb1
LHandIndex	mixamorig_LeftHandIndex1	LHandMiddle	mixamorig_LeftHandMiddle1
LHandRing	mixamorig_LeftHandRing1	LHandPinky	mixamorig_LeftHandPinky1
REye	mixamorig_RightEye	LEye	mixamorig_LeftEye

Figure 3: Bone Mapping from avatar to standard names.



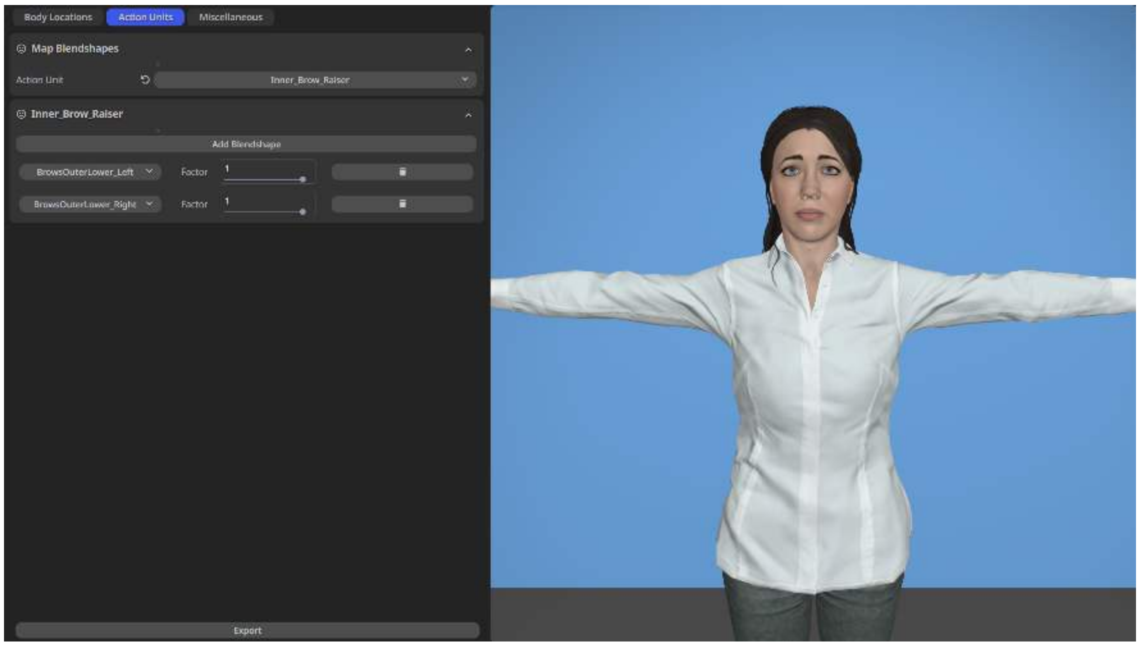
Select a point: shift + click
Commit changes to point: alt + click

Point: SHOULDER_LINE

Mode: Mesh Intersection | Translation | Rotation

Head	mixamorig_Head
Neck	mixamorig_Neck
ShouldersUnion	mixamorig_Spine2
Stomach	mixamorig_Spine1
BelowStomach	mixamorig_Spine
Hips	mixamorig_Hips
RShoulder	mixamorig_RightShoulder
RArm	mixamorig_RightArm
RElbow	mixamorig_RightForeArm
RWrist	mixamorig_RightHand
RHandThumb	mixamorig_RightHandThumb1
RHandIndex	mixamorig_RightHandIndex1

Figure 4: Body Locations editor.



Action Unit: Inner_Brow_Raiser

Inner_Brow_Raiser

BrowsOuterLower_Left	Factor	1
BrowsOuterLower_Right	Factor	1

Figure 5: Blendshape to Action Unit mapping editor.

In terms of implementation, Manual Features issues that might appear when changing avatar, are related mainly to the mesh and skeletal positions and proportions. The name of the bones might be different, but they can be easily mapped as long as their distribution is similar (Figure 3). Locations in the body, used to position the hands, are avatar-specific and thus need to be manually established and/or automatically computed. Although much of this process can be automated, a 100% success rate cannot be ensured and so the skilled user needs to check for and adjust any misalignment (Figure 4).

Regarding Non Manual Features, 3D models most often use blendshapes (BS) to represent multiple states of the same face. Also, a common practice for artists is to use the Facial Action Coding System (FACS)[4] to sculpt each BS. FACS allows to recreate nearly any anatomically possible facial expression, deconstructing it into the specific Action Units (AUs) and their temporal segments that produced the expression. We use a reference list of AUs in our 3D characters. However, users might use non-standard names for BS to represent an AU when creating an avatar. If an avatar is not modeled following the reference, does not accurately perform an AU or does not have a unique BS corresponding to an AU, the configuration file must contain a mapping of the avatar BS names to the reference AU list. Multiple BS, each with a weighting factor, can be provided to form a single AU (Figure 5). In addition, we also take care that the avatar face is composed of different meshes (such as the face, the tongue, the eyelids, the mouth, etc.) each containing BS that only alter some AUs. Therefore, a list with the specific AUs that affect each mesh can be provided, in order to avoid updating the BS of all the meshes when performing an expression.

The configuration file generation webpage can be used to create a set of different avatar options (or “skins”) to enable each user of the SignON SLMT App to personalise the avatar to their wishes, in a future operational version of the app, as the Performs realiser development results in the ability to seamlessly exchange the avatar.

[4] Ekman P, Friesen W. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto: Consulting Psychologists Press. 1978

4. Highest Quality Avatar Characteristics and Mobile Adaptation

In this section, we complement the details on the high quality avatar obtained within the framework of the SignON project, beyond what was discussed in D5.1, and discuss the techniques applied to adapt the 3D models for mobile devices.

4.1. Highest Quality Avatar

Within D5.1, we covered the technical aspects related to the skeleton structure of the avatar in order to perform SL animations, and we discussed insights of the rendering techniques of the body and eyes of the high-quality, virtual character we have been creating. The presentation of the final hair and blendshapes was incomplete, and will be discussed here.

4.1.1 Hair Details

As presented in D5.1, our hair approach is based on a hair card model. We redefined the design by reducing the length of the ponytail and the side hair strands that fell in front of the ears, and we also added a hair band. These changes were made anticipating possible limitations at simulating hair behaviour in mobile hardware, which will be discussed later on in this section. We also added a brown colour to the head mesh, which acts as scalp. Different views of the visual addition of hair in the head mesh (with and without material) can be seen in Figures 6 and 7.

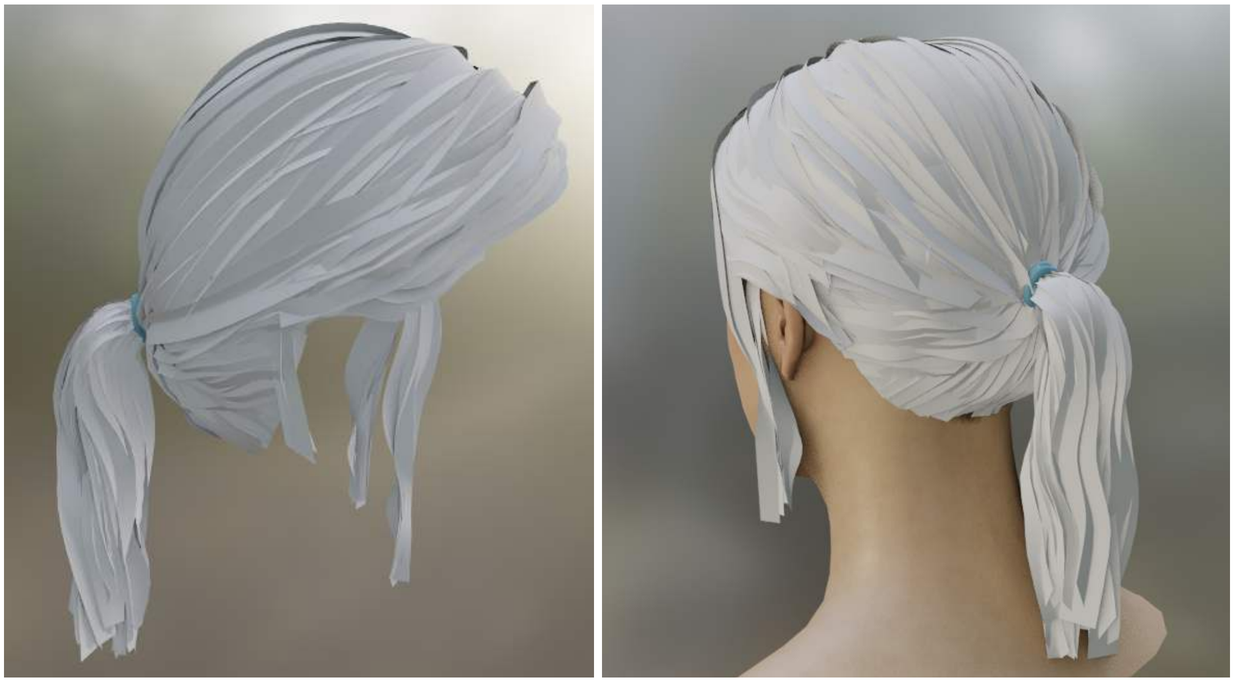


Figure 6: Different views of the hair cards model of the new hair.

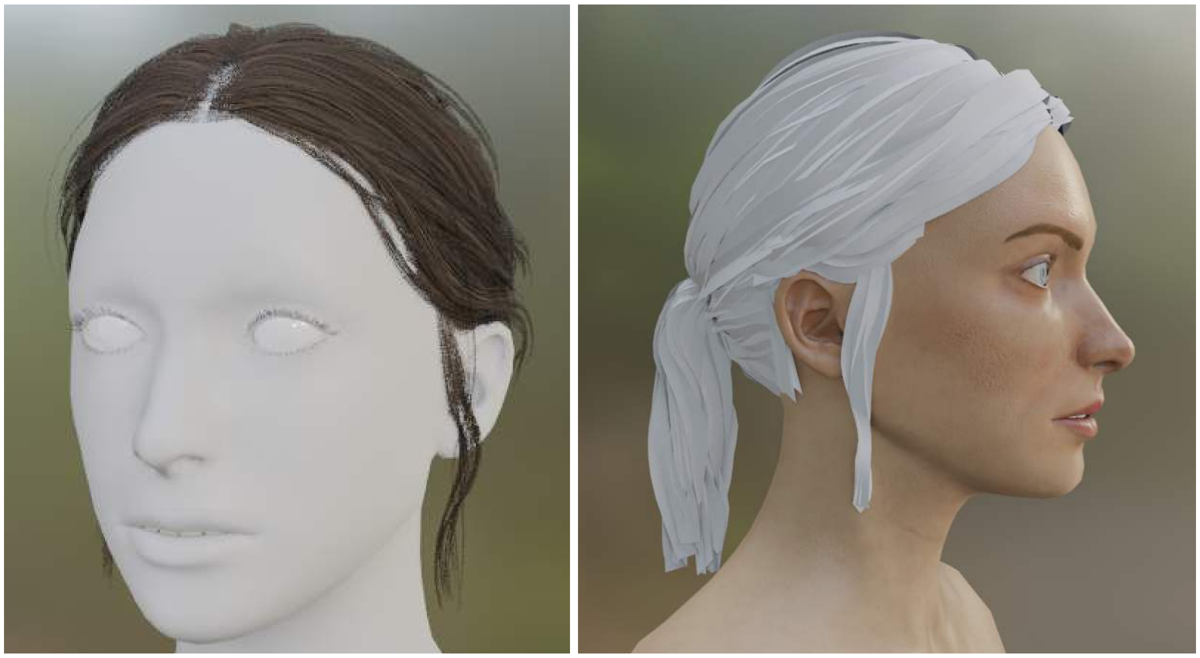


Figure 7: Different views of the hair cards model with and without the material applied.

In the same way it was made for the other meshes, a list of textures was also created in order to support Physically Based Rendering (PBR). As we can appreciate in Figure 8, the material enhances the realism and the quality of the final visual result. The textures created were *albedo*, *normals*, *opacity*, and *ambient occlusion* maps. We can see all of them in Figure 9.

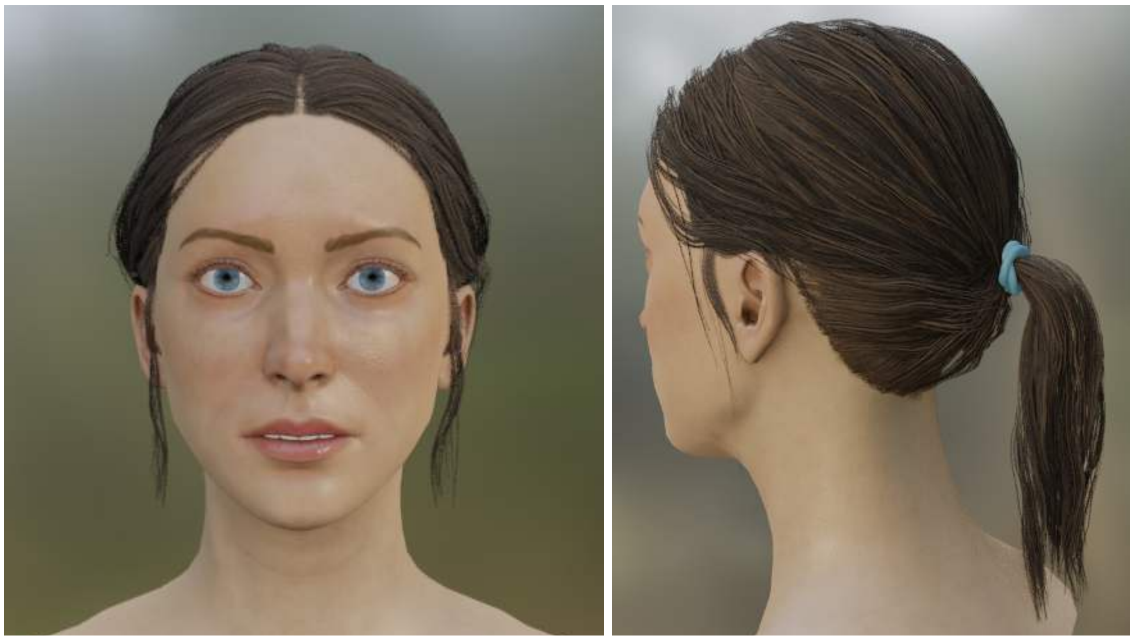


Figure 8: Final view of the hair mesh applied to the head mesh (all materials applied).

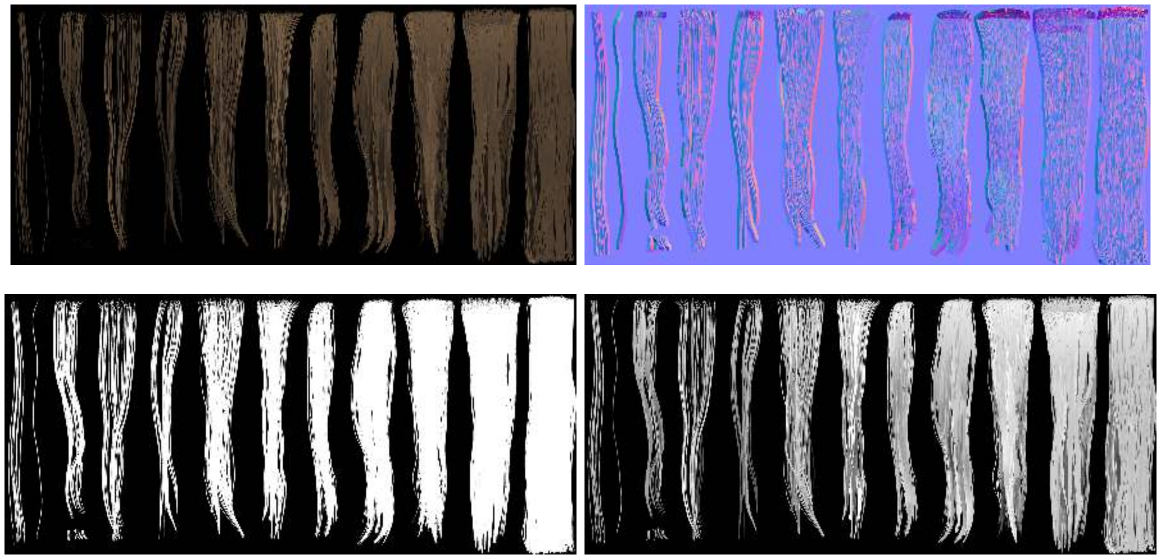


Figure 9: From top to bottom, from left to right: Albedo map, Normal map, Opacity map, and Ambient Occlusion map.

4.1.2 Blendshapes Details

Regarding the morph targets, in D5.1 we discussed the need of having a well-crafted list of BS to accurately support the NMFs. Furthermore, our previous model had errors in some BS used for mouthing (such as the tongue movement, or the kissing shape), causing that some face states could not be reproduced exactly as intended. All these problems are fixed in the new avatar, resulting into more natural poses (see Figure 10). The full list of the BS available for the new avatar can be found in Annex II. With them, we cover with much higher quality the widest range of facial expressions, which is one of the main requirements from users, as discussed in Section 2.

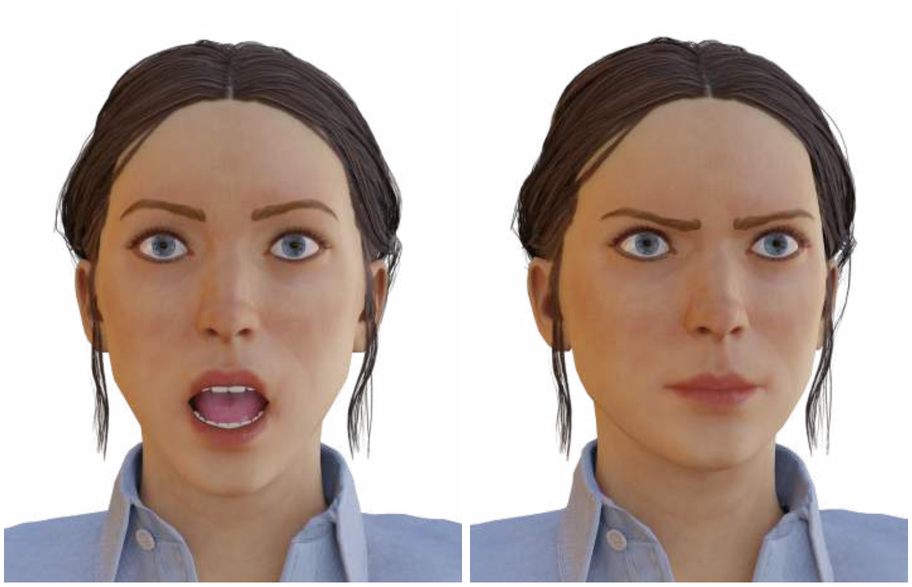


Figure 10: Right: Mouth_Stretch BS (intensity of 0.8) applied to the head, tongue, and lowerTeeth meshes. Left: Eyebrow_Frown_Left, Eyebrow_Frown_Right, and Mouth_Close_Down BS (intensity of 1.0) applied to the head and eyelashes meshes.

The integration of these different improvements results in a new avatar that looks better than the previous one used in SignON, as can be seen in Figures 11 and 12.

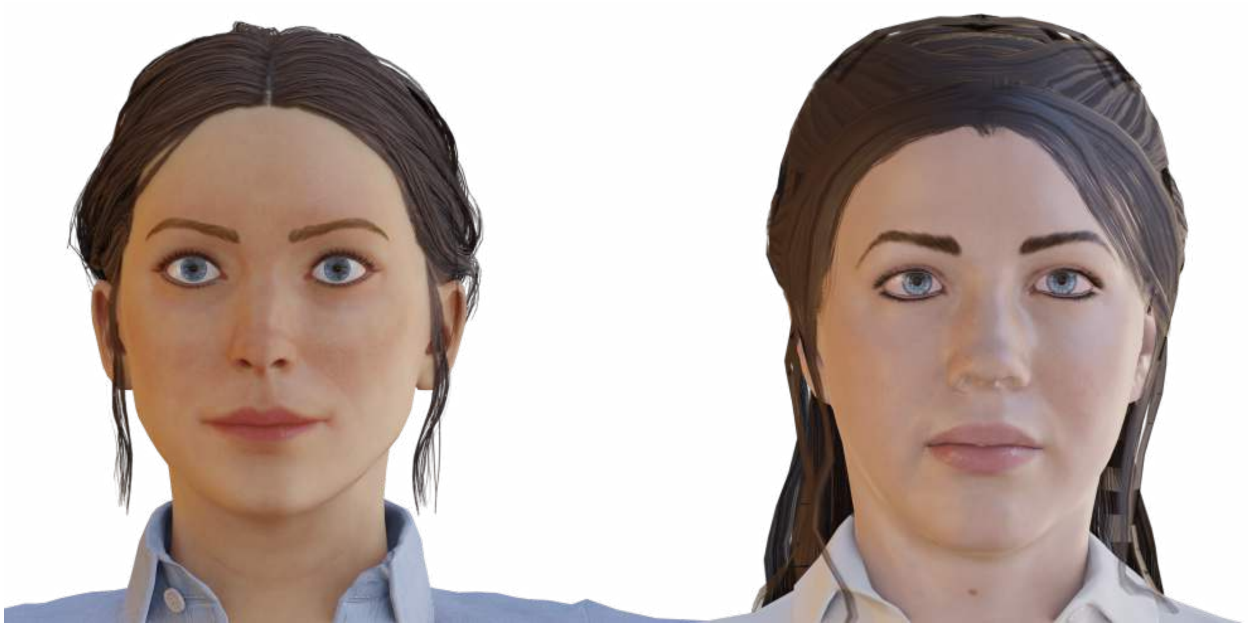


Figure 11: Side-to-side comparison of the heads between the new (left) and old (right) avatar.

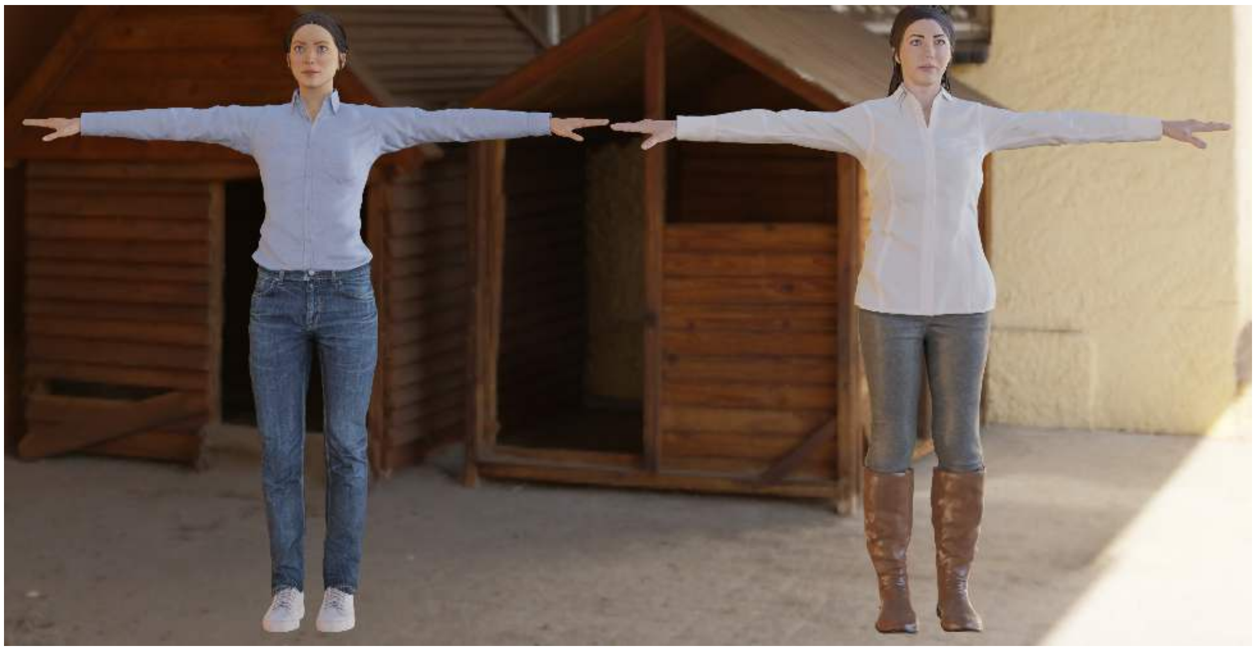


Figure 12: Side-to-side comparison of the whole body between the new (left) and old (right) avatar.

4.2. Mobile Adaptation

The new high-quality avatar will directly replace the previous avatar in our web applications (*Animics* and *Performs*). However, too much complexity is unaffordable for the mobile app due to the hardware/software limitations. Therefore, different modifications are needed to adapt the avatar to mobile use, such as not applying computationally expensive rendering techniques, simplifying rendering in parts that will not be noticeable in mobile screens; or to save memory and computation, not including meshes that will not be shown (shoes, body, legs), checking for repeated information (shape keys), reducing textures, and decimating the mesh. We discuss this further below.

From the rendering perspective, we have evaluated high quality rendering algorithms for eyes (by having an iris/sclera layer setup) and hair (Kajiya-Kay algorithm). Nevertheless, this ended up being an over engineered approach for mobiles where the computational cost is significant. As a consequence, simpler but more standard techniques such as PBR are used. On the other hand, some of the textures created for the new avatar will not be used in the mobile use case (e.g., the cavity map, the curvature map, or the thickness map).

Virtual characters are a combination of meshes that, once grouped, constitute the whole 3D model. Morph targets commonly only represent the different states of a very specific part of the body, but the object-oriented structure of 3D software sometimes spreads information into child objects or objects from the same parent. As a consequence, we found out that morph targets applied to the head mesh were also defined in the body mesh, making it a waste of memory usage. Deleting the duplicity of data helps us to decrease the file size considerably (around 10% in the case of some *Character Creator 4* avatars).

Character Creator 4 gives users the possibility to download dressed or undressed avatars. When they are dressed, the meshes representing the torso, legs, or feet are not visible, therefore they are a waste of computational resources. When the application of the 3D character is known, optimisation will remove non-visible meshes (with a 10% saving in our tests). A visual representation can be seen in Figure 13.

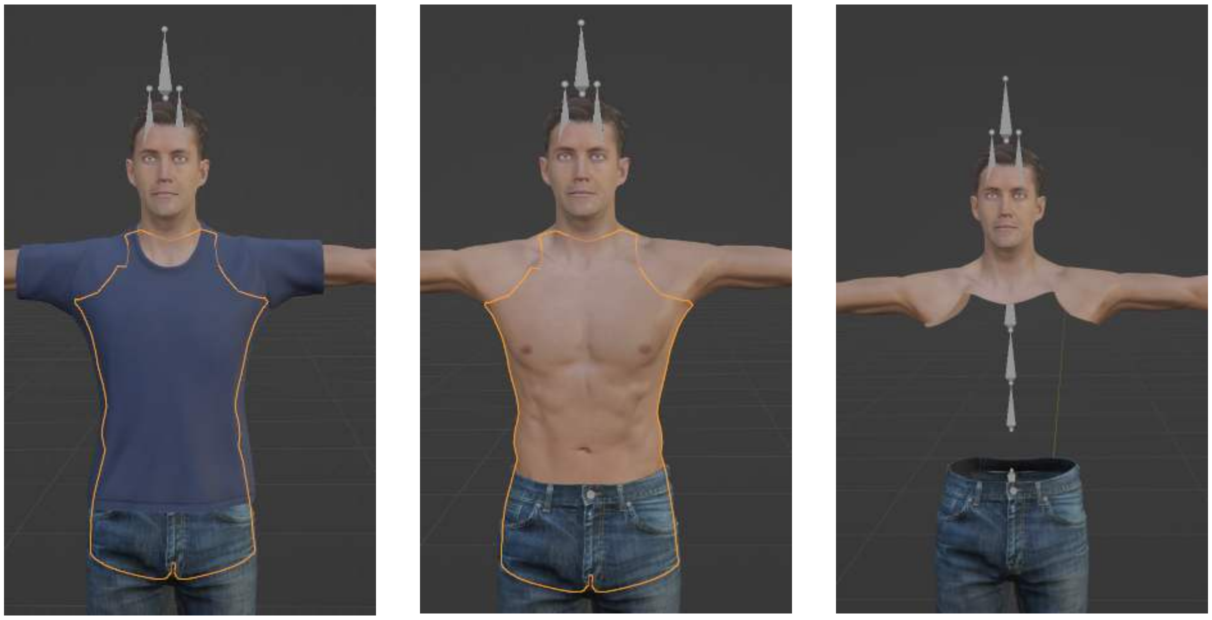


Figure 13: Process of removing the mesh that will not be visible once the 3D character is dressed.

High resolution textures are needed for higher end devices such as monitors or current TV sets. In the case of mobile phones, a downsampling can be applied to reduce the size of the file of the 3D model while also optimising rendering time. Specifically, we reduced the original texture resolution to a half without visual impact on the mobile app, decreasing the size up to 55% in some tests.

A final optimisation strategy, similar to the previous one, is related to the resolution of the meshes, which are composed of vertices. Very detailed ones are only considered when they can be clearly seen in the device. This optimisation process is called *decimating*. We applied a 40% vertices decimation without observing any visual difference nor deformation, which reduced the file size by 5% (indeed, reducing the vertex detail of meshes that are far away from the camera or that are rendered in a small part of the screen is a common practice).

In order to give a more cumulative picture of these changes, in our tests, a mesh with a size of 91.7 Mb, approximately, got reduced to one weighing some 17 Mb, without noticeable loss of quality in mobile devices. As indicated earlier, besides saving downloading time, these changes also decrease rendering computations.

5. Integration and Future Work

We have already tested the sign synthesis module using new characters, and the animations are correctly performed. As a short-term result, we expect to be able to show the animations based on the refactored system both with our own character, and one obtained and configured with the process explained in Section 3. In general, we believe this work to be a step towards setting a new state of the art in the quality of the avatars for procedural sign synthesis.

The SignON project demonstrated a commitment to user needs, a thoughtful approach to avatar customization, and a focus on maintaining interoperability. The sign language synthesis lessons we learned from our 3 years of R&D include the importance of standardized pipelines for avatar creation, addressing challenges in mobile adaptation, and continual refinement based on user feedback. We learned the importance of balancing visual realism with naturalness and accuracy in sign language synthesis, addressing NMFs effectively, developing scalable avatar generation pipelines, adapting avatars for mobile use, and fostering user involvement in avatar design. These lessons learned provide valuable insights for future advancements in sign language synthesis technology.

Our key sign language synthesis lessons learnt are:

1. Prioritise naturalness of sign language animations over visual realism: While users have differing preferences regarding avatar appearance, the consensus is that natural and accurate sign language animations should be prioritised over creating the most realistic-looking avatars. This aligns with the project focus on maximising the quality of sign animations.
2. Address non-manual features (NMFs) effectively: NMFs play a crucial role in sign language communication, conveying emotions and nuances that complement the manual signs. We found the importance of addressing NMFs effectively, ensuring that the avatars can accurately convey these non-verbal cues.
3. Develop a scalable pipeline for high-quality avatar generation: Creating a wide range of high-quality avatars is essential for supporting user diversity and preferences. We describe a pipeline that utilises Open Source models and software to generate diverse avatars efficiently.

4. Adapt avatars for mobile use: Mobile devices pose unique challenges due to hardware limitations. We developed strategies for adapting avatars for mobile use, optimising performance while maintaining visual quality.
5. Foster user involvement in avatar design: Collaborating with DHH users throughout the design process is crucial for ensuring that the avatars meet their needs and preferences. We cannot overemphasise the importance of user feedback and involvement in avatar development.

Annex I: Detailed Pipeline Guide

As indicated in Section 3.1, this Annex provides a more detailed guide of the avatar creation process. We start by providing a glossary of the main terms.

Armature or Skeleton	It represents the whole armature of the mesh. It is composed of a hierarchy of joints. We believe that armature is more common because it separates the dual meaning that skeleton suffers from.
Joint	The points of articulation of a character to control its movement. The movement is controlled by rotations of the joints.
Bone	While joints only represent a 3D point, bones also include a length by setting an end site to a joint.
Base Pose or Basis Pose	This term represents the state of the armature when all the joints are rotated towards their set of coordinates. It can be informally understood as if all the joints had no rotation applied.
Bind Pose or Rest Pose or Rest Position	It represents the state of the armature with an initial displacement and rotations for each joint. Basically, the pose in which the mesh was modeled. It is commonly represented with a T-Pose or an A-Pose.
Original Pose	This term is used in *Mixamo*. It represents the position in which the user uploads the avatar.
Pose Position	This term is used in *Blender*. It represents the pose of the first frame of the animation.

1) CC4 Block

A) *Create or load an avatar into the CC4 scene*
[TIP]In order to see the default content available in CC4 open the “Content Manager” (F4) from the menu window in the top bar

B) *Fix the shape keys*: some facial Expressions that are performed in CC4 such as Jaw_open, are a combination of bone animation + shape keys. Normally it’s more convenient to use just blendshapes for facial animation, in order to convert those animations into just blendshapes this process has to be followed.

For each facial Expression that we want to convert, set the facial expression slider to 100% and export the avatar from File>Export>OBJ >“Keep Facial Expressions in Nude Bind-Pose” with the default settings. Then on the same slider, click the pencil “Edit” button, a menu will appear, select the “File” option and load the previously exported .OBJ file with the facial pose (a checksum file will be automatically loaded too) and press the “OK” button.

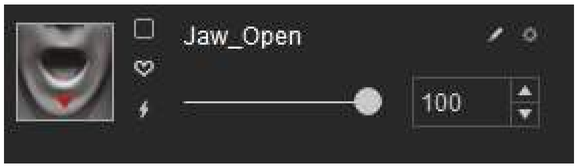
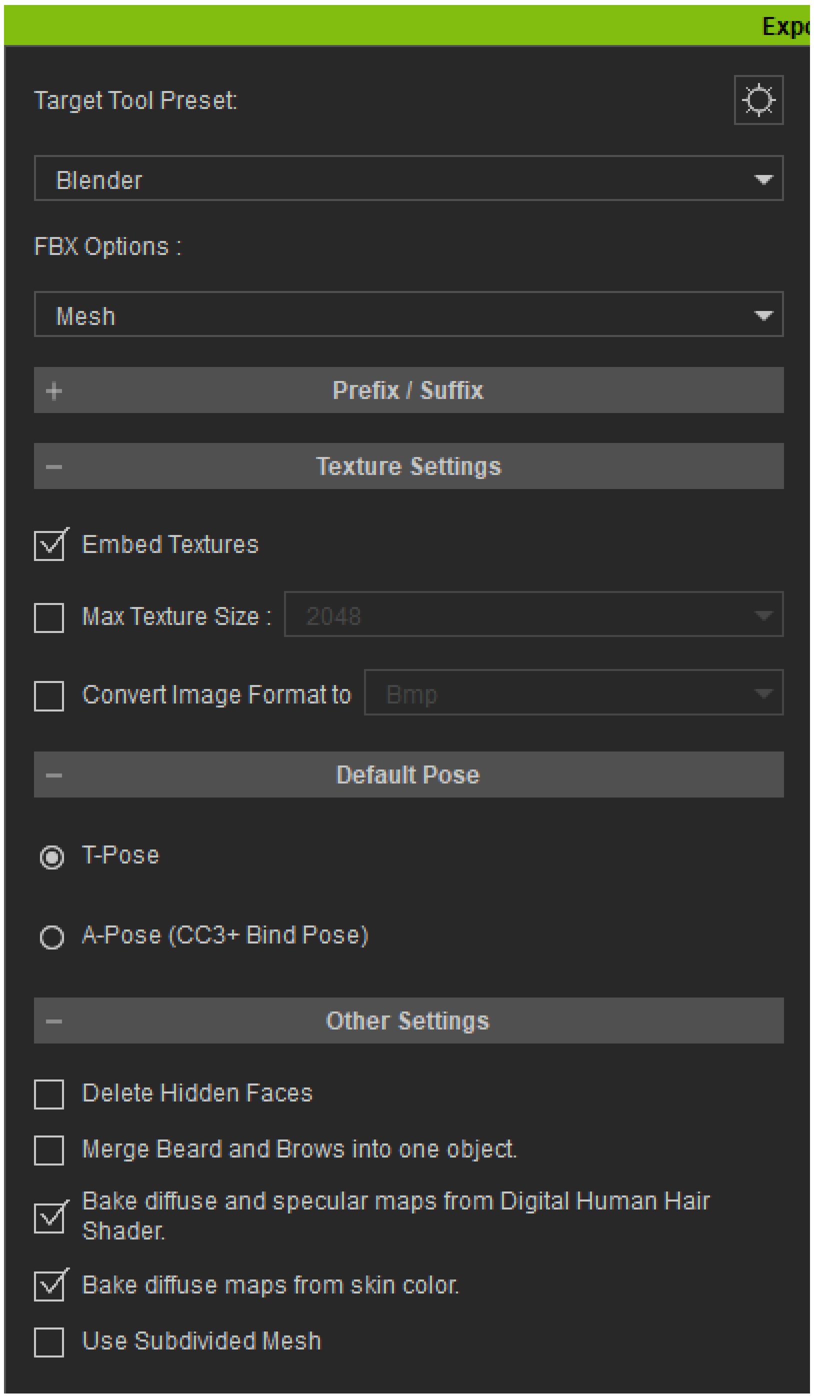


Figure 14: Widget for expression conversion

Once all the facial expressions have been fixed a “Facial profile” in the Content>Actor>FacialProfile can be saved in order to be reused with other avatars.

C) *Export the avatar*, to export it go to File>Export>FBX>Clothed Character and make sure to select as default pose T-pose and the Target tool preset is set to your desired tool, in our case “Blender”.



Expo

Target Tool Preset:

Blender

FBX Options :

Mesh

+ Prefix / Suffix

− Texture Settings

[✓] Embed Textures

[ ] Max Texture Size : 2048

[ ] Convert Image Format to Bmp

− Default Pose

● T-Pose

○ A-Pose (CC3+ Bind Pose)

− Other Settings

[ ] Delete Hidden Faces

[ ] Merge Beard and Brows into one object.

[✓] Bake diffuse and specular maps from Digital Human Hair Shader.

[✓] Bake diffuse maps from skin color.

[ ] Use Subdivided Mesh

Figure 15: Appropriate export settings

2) Blender Block (first pass)

A) *Load the avatar* .fbx into a blender scene

B) Select the "*Armature*" object and apply all the transformations to it with Object>Apply>All transforms or Ctrl+A>All transforms

Remove all the shape keys for every object inside the armature. This is done by clicking the object, going into the "Data" attribute of the object

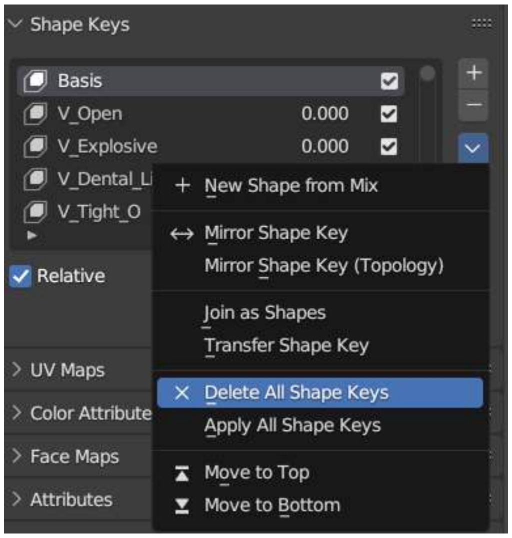


Figure 16: Removing shape keys

C) Select the Armature and go to the "Data" section, there are two pose options "Pose position" and "Rest position". Take a look at what objects are changing positions between poses, *those objects that change* (normally, the arms, torso and torso clothes) *need to be corrected*. To do so, do the following process on those objects:

1) Leave the pose position set
2) Select the object and go to "Modifiers" (Wrench icon)
3) Duplicate the "Armature" modifier (you'll see how the avatar mesh is displaced). A new modifier named Armature.001 will be created below

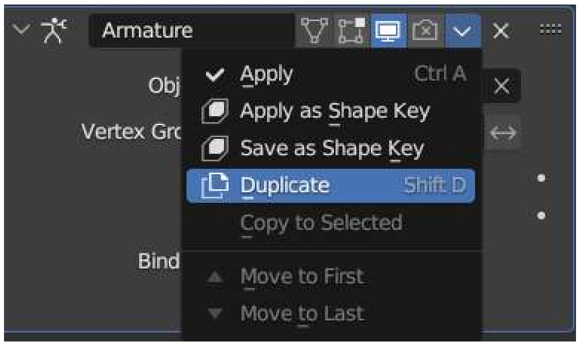


Figure 17: Armature duplication

4) Apply the first Armature

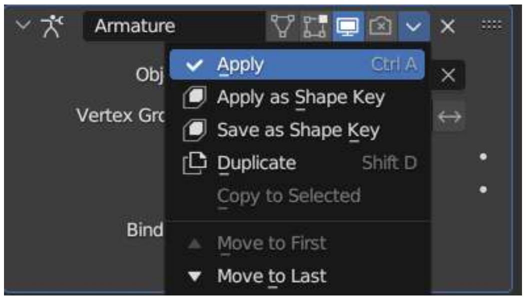


Figure 18: Applying armature

When this is done for all the needed objects, select the Armature object (not the modifier) and switch to pose mode, once in pose mode.

In pose mode go to Pose>Apply>Apply Pose as Rest Pose

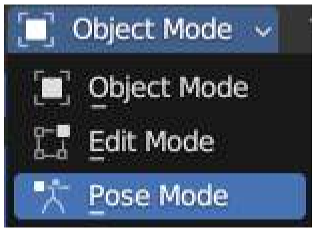


Figure 19: Applying rest pose

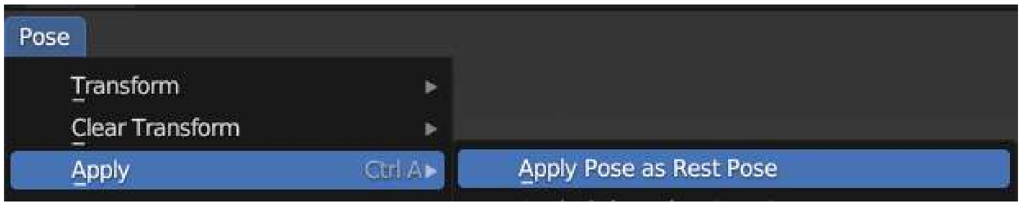


Figure 20: Second stage to apply rest pose

D) Now you should end up with your Avatar in the exact same T-pose when selecting both the “Pose position” and “Rest position”

-Delete the “Armature” object

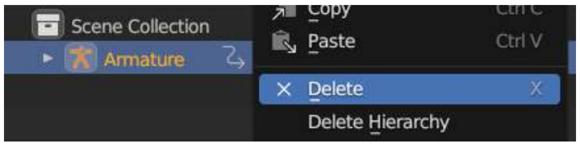


Figure 21: Deleting armature

E) Select all the resulting objects with shift+click and apply all the transformations to it with Object>Apply>All transforms or Ctrl+A>All transforms

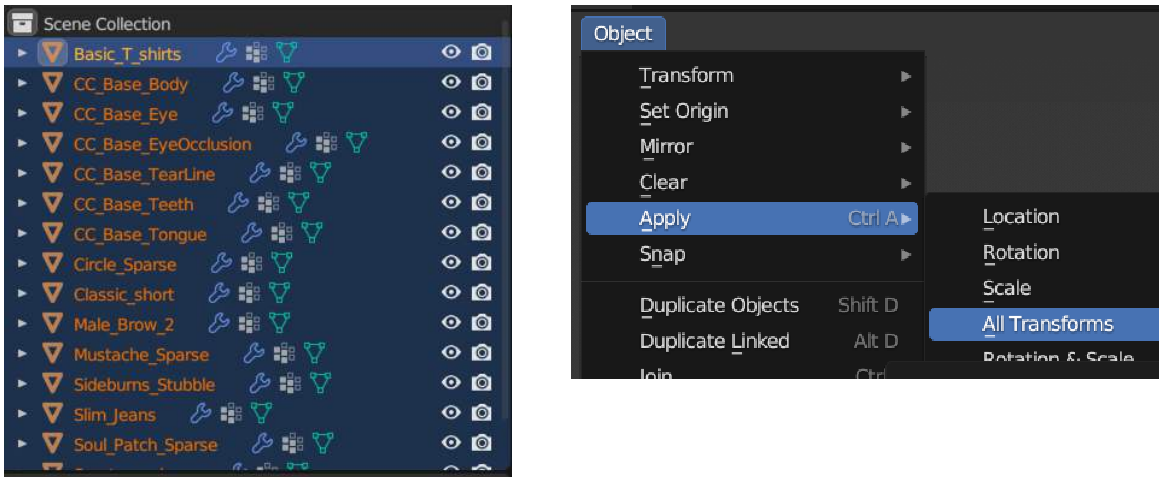


Figure 22: Final stages of the process

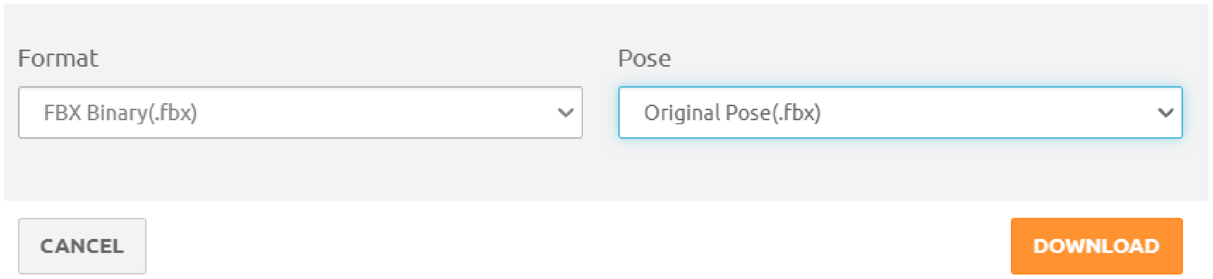
F) Export the avatar as an FBX

3) Mixamo Block

A) Upload the FBX to mixamo

B) Perform the rig

C) Download the rigged avatar with the Original pose


Format: FBX Binary(.fbx)

Pose: Original Pose(.fbx)

Figure 23: Downloading widget

4) Blender Block (second pass)

A) Import the Mixamo rigged character into the scene, in the blender scene this will be the object "Armature"

B) Import the original FBX from CC4, in the blender scene this will be the object "Armature.001"

C) Select the "Armature.001" object and go into pose mode, then go to Pose>Apply>Apply Pose as Rest Pose. (The avatar objects will change position but this does not affect the final outcome as the important part is the skeleton bones and their position)

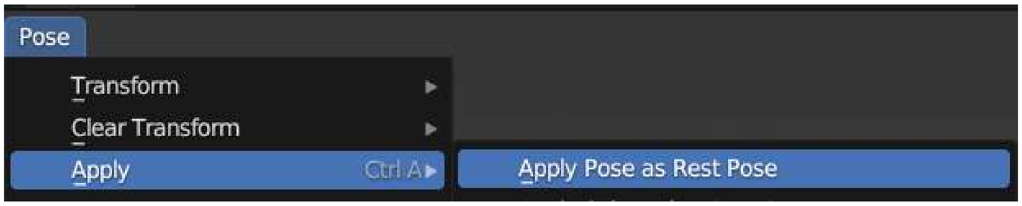


Figure 24: Setting correctly the pose

D) Navigate to the "scripting" tab on the top bar menu, and with the button "Open" import the "script.py" file. (Documentation on the script provided below)

E) Execute the file and wait until it finishes (it might take a while).

F) Export the completed avatar as a glb from File>Export>glTF 2.0

Script explanation

The script is divided into various steps:

1) Fix and Merge Armature:

- Applies transforms to all objects in the scene and sets all objects to visible
- Transfers shape keys from one armature to another, and removes meshes from armature_CC4.

2) Armature Cleanup:

- Delete hands bones on the mixamo armature
- Delete all bones but the hand ones on the cc4 armature

3) Skeleton Join:

- Joins two armatures, armature_CC4 and armature_mixamo,and parents hands bones.

4) Add Fingers & Rename:

- Renames specific bones.
- Modifies the armature and creates additional bones for fingers (endsites)

5) Correct Materials:

- Updates materials to ensure they have the correct textures, shaders and naming.

6) Rigging:

- Copies the body mesh, merges its vertices, and applies automatic rigging to it.
- Transfers weights between the copied body and other objects (clothing).
- Merges vertices of all objects.

7) Reduce Blendshapes:

- Removes shape keys (blendshapes) from objects, except for face-related shape keys.
- Separates parts of the mesh based on materials and renames the objects.

8) Eye Bones & Rename Armature:

- Renames bones in the armature and creates eye bones.
- Assigns vertex groups for the eyes and positions the eye bones.

9) Fix skeleton position

- Align the arms positions to point at a the desired vector (1,0,0).
- Rotate the bone rolls so all bones in the arm have the same orientation XYZ.
- Modify the hand bone to be aligned with the middle finger.
- Copy the arm bone rotation to all his children.
- Align the hand bone with the middle finger.
- Apply all the transformations to the bind pose.

On Poses

Because of technical simplicity with respect to the rotations interpretation, some base poses are as shown on the figures that follow. They are quite difficult for users to perform.

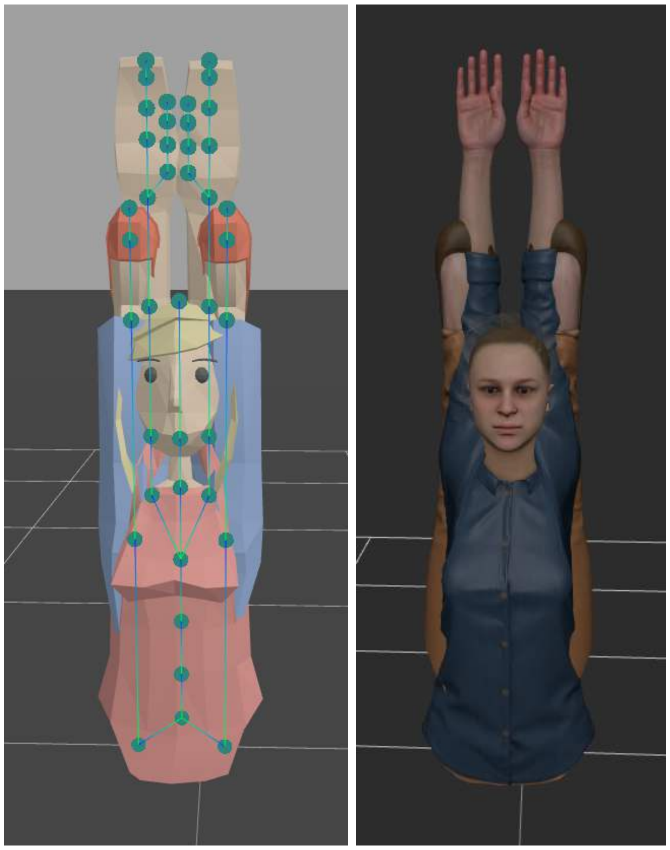


Figure 25: Illustration of multiple pose settings

As quite a few animations are based on Motion Capture (MoCap), the initial position is a more natural user position, the T-pose, shown on the image next. As its use is quite extensive, we have decided to use this as the reference pose, the standard one for our system. We need to make sure to convert the initial pose of any avatar to this reference pose.

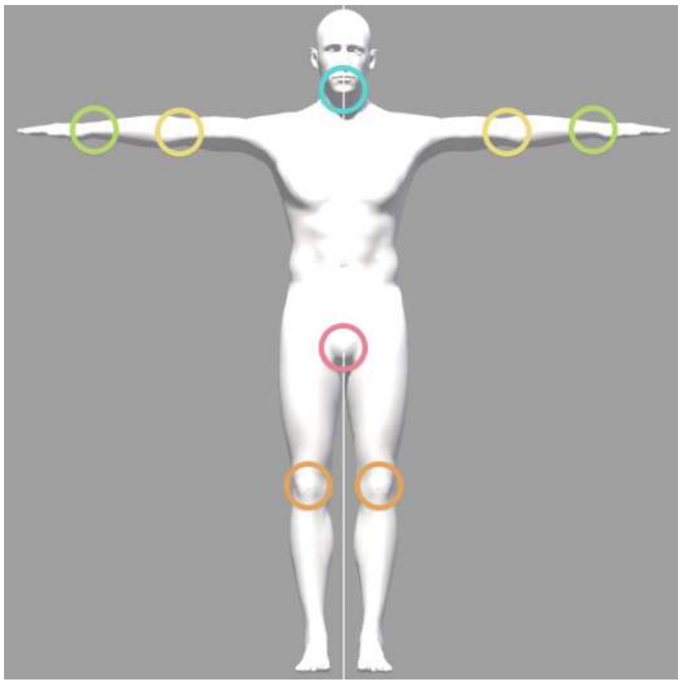


Figure 26: Additional example of pose

Annex II: List of Available Blendshapes

As indicated in Section 4, this Annex provides the full list of blendshapes created for the different meshes of the new 3D character.

Mesh: Head.

Body Part	Lexeme (Name)	Reference Action Unit
Eyebrows	Eyebrow_Frown_Left	AU4
Eyebrows	Eyebrow_Frown_Right	AU4
Eyebrows	Eyebrow_Arch_Left	AU1 + AU2
Eyebrows	Eyebrow_Arch_Right	AU1 + AU2
Eyes	Eye_Open_Left	AU5
Eyes	Eye_Open_Right	AU5
Eyes	Eye_Squint_Left	AU44
Eyes	Eye_Squint_Right	AU44
Eyes	Eye_Blink_Left	AU45
Eyes	Eye_Blink_Right	AU45
Cheeks	Cheek_Suck_Left	AU35
Cheeks	Cheek_Suck_Right	AU35
Cheeks	Cheek_Blow_Left	AU33
Cheeks	Cheek_Blow_Right	AU33
Mouth	Mouth_Stretch	AU27
Mouth	Mouth_Close_Up	AU24
Mouth	Mouth_Close_Down	AU24
Mouth	Lip_Dimpler_Left	AU14
Mouth	Lip_Dimpler_Right	AU14
Mouth	Lip_Apart	AU25

Mesh: EyeLashes.

Body Part	Lexeme (Name)	Reference Action Unit
Eyes	Eye_Open_Left	AU5
Eyes	Eye_Open_Right	AU5
Eyes	Eye_Squint_Left	AU44
Eyes	Eye_Squint_Right	AU44
Eyes	Eye_Blink_Left	AU45
Eyes	Eye_Blink_Right	AU45

Mesh: LowerTeeth.

Body Part	Lexeme (Name)	Reference Action Unit
Mouth	Mouth_Stretch	AU27
Tongue	Tongue_Point	AU19

Mesh: Tongue.

Body Part	Lexeme (Name)	Reference Action Unit
Mouth	Mouth_Stretch	AU27
Tongue	Tongue_Point	AU19
Tongue	Tongue_Wide	None. Mouthing specific.
Tongue	Tongue_Front_Up	None. Mouthing specific.