Facerig Model and Textures Documentation – Importer V2

v.01 (work in progress)

Please note that script that prepare your data for Facerig may take a while (1-2 minutes) to finish the job. The duration of the process depends on how big your model and textures are and on how powerfull your computer is.

At the moment the script doesn't work with file paths that contains white spaces so it's strongly recommended to remove the spaces.

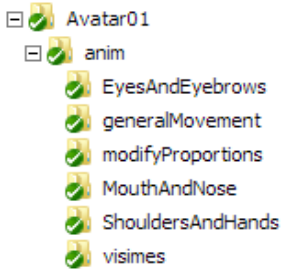
What's new

This new version of the importer script differs from the previous one by **the way it handles animations.** The previous version did some **processing** to the user animations in order to prepare them for Facerig. That processing **reduced the number** of animations required from the artist and exported the animation only on **specific bones**, lifting that requirement from the user. However this behavior was relying on certain bone names and specific animation relations that could not be update with ease.

The new importer script, require animations as Facerig needs them but give liberty on **choosing the bone hierarchy** and enable us to **keep up to date the features** exposed to the community artists. This time the discipline is moved upon the way animations are named, placed in subfolders and what bones give specific offsets in each animation.

As with the previous version of the importer the data is mainly required as **collada** and **targa** files. But unlike that version it's organized differently.

The file holding the geometry and skeleton and the texture files must be placed in the same directory, same as before. The artist will be prompted with a browse window in order to specify this data folder. However the animations will be placed in a subfolder structure placed in this base directory. Below you can see the folder structure of Fluffo.



More about animations later in the dedicated section of the documentation.

There is an aditional **txt** file that can be placed near the collada and if present this file will be pick-up by the script and transformed in the **configuration file** that sets various avatar parameters. The configuration file is placed by the script in "SteamApps\common\FaceRig\Mod\VP\PC_CustomData\Objects**YourAvatar**". You can further modify this generated file directly. Its extension is .cfg and has a "cc" prefix (CustomConfiguration).

You can find more information on this below in “Avatar configuration file” section.

Geometry

- The geometry should be exported as a collada file with the model in **default pose**.

Geometry Naming

The file that holds the geometry must be named the following way: name + “Geometry”, where the name is the desired name of the avatar and “Geometry” is a string indicating that the file is delivering geometric information not animation. For instance if the **avatar** should be named “Fluffo”, the geometry file should be named “**FluffoGeometry**”.

If the model is a **prop** for an existing avatar then the name should be structured like this: avatarName+”_”+prop+”_”+propName + “Geometry”. For instance if the prop is meant to work with avatar Fluffo and the prop is named “vikingHelmet”, the geometry file should be named “**Fluffo_prop_vikingHelmetGeometry**”.

If you want to make a **generic prop**, available for all avatars then the name should be like this: “_prop_”+ propName +“Geometry” (for instance it could be “_prop_romanHelemetGeometry”)

If your folder data doesn't contain any file for geometry export the processing script **will abort the model preparations** and the model will **NOT** be integrated in Facerig.

Default pose

In the default pose deactivating skinning should NOT produce any changes in model appearance. If it does then the pose is not the default one.

Geometry rules

- All geometry must be polygonal (not procedural, NURBS or any other kind).
- Separate shells/elements (non connected surfaces) are permitted. Open surface is permitted.
- The polygon count is recommended to be under 50000 triangles. If the model is lighter that's even better. On powerful PCs the polycount can be greater but otherwise highpoly count will affect the framerate.
- For a sense of scale, the height of the avatar should be around 2 units. It doesn't matter what that units represents, inch, centimeters, generic units etc.
- the polygonal surfaces must have the transformation reseted to identity. That means their axes should be aligned to the world axes, the position should be 0,0,0 and the scale should be neutral (1,1,1 or 100,100,100 depending of the software defaults).
- all the surfaces must be skinned.
- the model should use a maximum of 10 materials.

UV Mapping

- The model should have normalized mapping (in 0-1 UV space). It's possible to have overlapping mapping faces but we encourage unique mapped models (with no overlapping) because there are shader types that need this kind of mapping, like the ones that have subsurface scattering.
- all polygons must have mapping coordinates.

Deformation

All deformations are done via skeletal deformation (skinning). No blendshapes, displacement or any other kind of deformation is supported.
- The number of bone affecting the geometry (via skinning) should be under 100.
- the skinning influences per vertex should be 4 or less. Anything above will be disregarded.
- skinning interpolation must be linear, not quaternion or other kind.

Fur

The fur is controlled by **both** geometric information and texture maps. In this section are explained the parameters controlled by the geometry.
The polygonal mesh can deliver fur information via **vertex color**. The direction is given much like a normal map, with red and green channel denoting a **tangent space angle**. Additionally the alpha channel in the vertex color denotes the **local length ratio**.
The **absolute length** of the fur is driven by the shader. From this length the vertex alpha channel can dictate a ratio for strands shorter than that maximum value. A value of 255 would allow the fur to be at maximum value. A value of 128 would set the fur to half of the maximum length, while a value of 0 would set the fur to 0 length, or no fur.
Because the direction of the fur is expressed in tangent space and Facerig uses a custom tangent space, **the actual colors will be written in vertices in the import process**. In order to do that the script needs **pairs of nodes** that represents hair root and hair tip. This nodes should be named “hairRoot” and “hairTip” repectively, followed by a node index, something like “hairRoot_01” and “hairTip_01”. The tip node **must** be linked by the root node.
The position of the hairRoot and hairTip will be used in the calculation, so ensure that those positions correctly represent the fur combing you are trying to achieve. Hair root should be as close as possible to the surface that will have fur.
Each surface meant to have fur info must be named “BlendedMesh” followed by an index, something like “BlendedMesh_01”. This also holds for rigid meshes, **the name is used to identify the surfaces that should have fur color information**.
The hair nodes will be removed from the model after the color information is written in vertices for not to clutter the file holding the skeleton.

Shader (materials) naming

The name of the materials should be made following this structure:
modelName+_shaderType+ shaderName .

For instance for an avatar named “Fluffo”, and a material named “Face” the shader name will be **“Fluffo_sht_furnormals_face”**
where “Fluffo” is the model name, “_sht_furnormals” is the shader type name, and “face” is the shader name.

If you want to have transparency in your shader, its names should contain:

-”blend1” for alpha blending
-”blend2” for alpha test
For instance for making a transparent bandana the shader could be named:
“Fluffo_sht_metalcloth_bandana_**blend2**”.

For information regarding shader types please see “Texures and Shaders Guidelines” documentation.
All shader names must by **lower case,** even though the avatar name is otherwise with first letter capitalized. That means that even though the avatar name is Fluffo (with “F”) **in the shader name** will be “fluffo” (with “f”).

For how to make **additional skins** please see “Textures maps” section.

Shader Types:

1. “sht_furnormals”: it's meant to be used on furred surfaces. It uses two normal maps, one that serves as a base with always present information, and one that can be activated in certain situations, for instance in frowning expression. In total this type of shader uses 7 textures and vertex color information in all four channels.
The RGB vertex color is used to specify the direction in which the fur is growing (similar to how a normal map would indicate direction) and alpha channel info gives the fur length (0 value means zero length fur, 255 value for full fur length). This normalized length value is a ratio from a length parameter exposed in the shader which the absolute value of the longest fur.
Textures used by this type of shader: diffuse (“d”), normal (“b”), normal 1 (“b1”), specular(“s”), fur mask (“m”), ambient occlusion (“ao”), colorize mask (“cm”).

2.“sht_furnonormals" Has the same characteristics as “sht_furnormals" except the animated normals feature, uses only one normal map texture.

3.“sht_physfurnormals" Has the same characteristics as “sht_furnormals" and features a physics determinate movement of the fur.

4.sht_metalcloth: Shader type targeted towards metal and cloth material representation. Textures used by this type of shader: diffuse (“d”), normal (“b”), normal (“b1”), normal (“b2”), normal (“b3”), normal (“b4”), specular(“s”), specular2 (“s2”), ambient occlusion (“ao”), colorize mask (“cm”).
The ambient occlusion channels have information corresponding with normal maps textures:

Red channel: ao corespondent to normal map nr. 1
Green channel: ao corespondent to normal map nr. 2
Blue channel: ao corespondent to normal map nr. 3
Alpha channel: ao corespondent to default normal map.

5.“sht_eye": Textures used by this type of shader: diffuse (“d”), normal (“b”), normal 1 (“b1”), specular(“s”), iris texture(”ristext_d”), (“raotext_d”).
The iris texture:”ristext_d” Used for iris phenomena representation and properties. The alpha channel holds the distance from the outer shape to the iris. Should be 0 outside the iris area and 255 in the center of the iris area resulting gradient falloff should describe a dome.
The RGB (grayscale) defines the area affected by Pupil Dilation. 0 No effect, 255 max effect. Should be 0 outside the iris and 255 throughout the black area in the middle of the pupil. The gradient variation between that 0 and that 255 controls how realistic the iris stretch looks.
The “raotext_d” texture represents environment reflectivity and specular sharp occlusion mask and

colorized ambient occlusion.

6. “sht_teethblendn” Shader used for the interior of the mouth. Textures used by this type of shader: diffuse (“d”), normal (“b”), specular(“s”), ambient occlusion (“ao”), colorize mask (“cm”), subsurface scatter (“sc”).
The specular texture is interpreted like this:

- the red channel gives information about the Glossiness factor >1 for rough surfaces 255 for shiny ones. Between 1 to 5 for rough surfaces, 5 to 12 for a very little shine and increasingly shiner towards 255.
- the green channel: ambient occlusion for fully closed mouth.
- the blue channel: Is used to specify areas of object that remains unaffected by lighting resulting in a light emitting effect when placed in the shadow (glow in the dark effect), 0 for non emissive and 255 for full emissivity.
- the alpha channel: Gives information about the Specular Power 0 no specular 255 full specular.

7. “sht_hair" diffuse (“d”), normal (“b”), specular(“s”), ambient occlusion (“ao”)
The red channel of the specular texture is not used, it should be set at 255 default value.
The ambient occlusion textur is interpreted like this:

Red channel: Not used, 255 default value.
Green channel: Direct light Attenuation 0 means no direct light (only ambient and radiosity) 255 full direct light intensity.
Blue channel: Not used, 255 default value.
Alpha channel: Base PPAO

8. “sht_skinblendn" Especially designed for skin areas of the model. Textures used by this type of shader: diffuse (“d”), normal (“b”), normal (“b1”), normal (“b2”), normal (“b3”), normal (“b4”), specular(“s”), ambient occlusion (“ao”), colorize mask (“cm”), subsurface scatter (“sc”).
There should be one default normal map texture and 4 additional normal map textures used for animated normals.
The ambient occlusion channels have information corresponding with normal maps textures:

Red channel: ao corespondent to normal map nr. 1
Green channel: ao corespondent to normal map nr. 2
Blue channel: ao corespondent to normal map nr. 3
Alpha channel: ao corespondent to default normal map.

The specular texture is interpreted like this:

Red channel gives information about the Glossiness factor >1 for rough surfaces 255 for shiny ones. Between 1 to 5 for rough surfaces, 5 to 12 for a very little shine and increasingly shiner towards 255.
Green channel: PPAO corespondent to normal map nr. 4.
Blue channel: Is used to specify areas of object that remains unaffected by lighting resulting in a light emitting effect when placed in the shadow (glow in the dark effect), 0 for non emissive and 255 for full emissivity.
Alpha channel: Gives information about the Specular Power 0 no specular 255 full specular.

The scatter texture is used like this:

Red channel: Range of sub surface scattering 0 for no sub surface scattering 255 max sub surface scattering. Should fade towards 0 on mapping seams and on solid details such as

piercings/jewelry.

Green channel: Direct light Attenuation 0 means no direct light (only ambient and radiosity) 255 full direct light intensity.

Blue channel: Amount of diffuse light that scatters in the sub-surface vs the amount of light that bounces right off. 0 means all light bounces right 255 all light scatters. A good default value would be 0.9 (229 229 229) for fleshy areas. Should fade towards 0 on mapping seams and on solid details such as piercings/jewelry.

Alpha channel: Translucency intensity (ears, nostrils) 1 for no translucency 0 for max translucency.

9.“sht_skinnoan" Has the same characteristics as “sht_skinblendn" except the animated normals feature, it uses one normal map texture.

10.“sht_prelit" uses 1 texture and vertex color information. It is needed to map a small geometry ring around the eye globes and eye lids to simulate eye shadow.
1. Diffuse texture: Small 64/64 pixels texture with RGB value of 128 and 255 value in alpha.
It uses alpha vertex color values for transparency, 255 opaque 0 full transparency , and RGB values for overall color.

Texture maps

Textures used by shaders are bitmaps in *.tga file format with alpha channel. Texture size should be **power of two** (16/16,32/32,64/64,128/128,256/256,512/512,1024/1024 pixels etc.).

avatarName+”_”+shaderName+”_”+textureType.tga
avatarName = avatar name you consider appropriate.
shaderName = avatarBodyPart name ex. :body, head, arms, goggles, Etc. the same used to name the material on the 3D model.
textureType = suffix denoting diffuse, normal map etc (see Texures and Shaders Guidelines)

For instance a texture named
“fluffo_face_d.tga” represents:
fluffo – avatarName
face – shader name
d – texture type, in this case diffuse texture

Texture Types Suffixes

“d” for diffuse textures.

“b” for normal map textures.

“b1” for additional animated normal map textures

“b2” for additional animated normal map textures

“b3” for additional animated normal map textures

“b4” for additional animated normal map textures

“s” for specular textures.

“s2” additional specular texture.

“sc” for skin scatter textures.

“ao” for ambient occlusion textures.

“m” for fur mask textures

“cm” for color mask textures

“iristext_d” used only by “sht_eye” template for iris phenomena representation and properties.

“raotext_d” used only by “sht_eye” template to mimic a faked ambient and specular occlusion on the eye ball.



1. Diffuse texture:”d”

Represents the diffuse color of the object stored in RGB channels. Alpha channel is used to specify transparency, 0(black)areas for full transparency and towards 255 (white) surfaces get increasingly opaque.

2. Normal map textures: “b”

These textures are expressed in a specific tangent space and for proper usege should be produced by Holotech's tools available for this operation. More information on how to produce compliant normal maps in

“NormalMapsForFacerig_01.pdf”



3. Specular texture:”s”

RGB



ALPHA CHANNEL



Each RGB channel has different information.

Red channel gives information about the Glossiness factor >1 for rough surfaces 255 for shiny ones. Between 1 to 5 for rough surfaces, 5 to 12 for a very little shine and increasingly shiner towards 255.

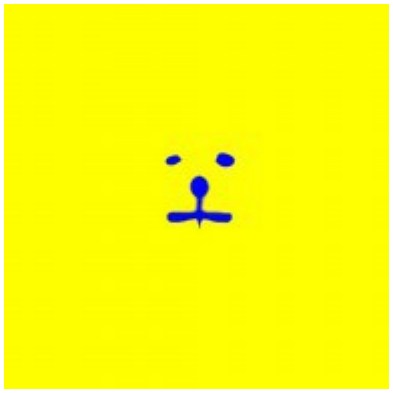
Green channel used to set values for different materials, from 0 for hair and cloth, to 255 for other material types (metal, plastic, etc).

Blue channel: Emissive channel, is used to specify areas of object that remains unaffected by lighting resulting in a light emitting effect when placed in the shadow (glow in the dark effect), 0 for non emissive and 255 for full emissivity.

Alpha channel gives information about the Specular Power 0 for no specular 255 full specular.

4. Fur mask texture:"m"

RGB



ALPHA CHANNEL



Red channel: Skin specular behavior(anisotropy). 255 values mean skin specular behaves like very short fur or hair. 0 values mean skin specular behaves like normal specular.

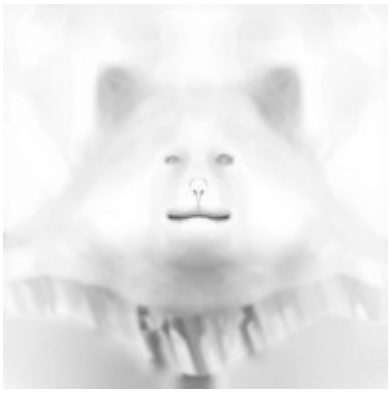
Green channel: Fur or hair transparency. We want the fur or hair to be transparent to see skin in areas with a lot of information from normal maps, such as the face. 255 for full fur or hair,0 for no fur or hair.

Blue channel: Skin specular intensity.0 for no specular on skin (recommended for fur or hair areas), 255 for full specular on skin (good for skin, tongue, teeth, nails).

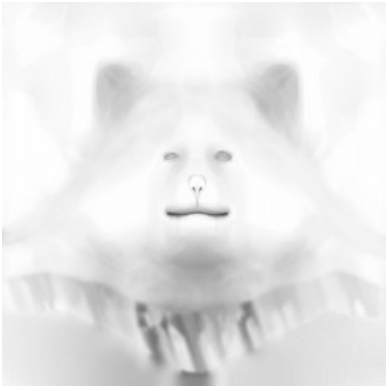
Alpha channel: fur or hair clump distribution, 0 values for short hair and 255 for long hair.

5. Ambient Occlusion texture:"ao"

RGB



ALPHA CHANNEL



Red channel should hold the PPAO(Per Pixel Ambient Occlusion) information for tensed muscles.
Green channel should hold the directional attenuation information for relaxed muscles.
Direct light Attenuation 0 means no direct light (only ambient and radiosity) 255 full direct light intensity.
Blue channel should hold the directional attenuation information for tensed muscles.
Direct light Attenuation 0 means no direct light (only ambient and radiosity) 255 full direct light intensity.
Alpha channel should hold the PPAO information for relaxed muscles.

6. Colorize mask texture:"cm"

RGB



ALPHA CHANNEL



For color customization, a texture file is used to hold information about which areas of the model receives new colors. Texture areas filled with 0 (black) will allow color change and those filled with 255(white) will reject color change. Areas ranging from 0 to 255 will also allow color change but it will be more obvious as they are closer to 0 value.
You can use RGB and Alpha channel and have 2 distinct color option and complex patterns.

Additional skins for avatars and props

If you want to make additional skins for the same shader at least one of the textures used by that shader should contain in its name "_sk"+index.
For instance if you want add another color for bandana, one of its textures, let's say the diffuse texture should have AN ADDITIONAL version named something like "Fluffo_bandana**_sk2**_d". If the skin should have transparency the name should be "Fluffo_bandana_blend1_sk2_d", where "blend1" denote alpha blending and "sk2" denotes a second skin for the shader.

Skeleton

As said in the introductory part is now possible to use **custom hierarchies and bone names**. The only bone names that are **still required** are: "Camera", "BipHead", "BipLEye" and "BipREye". These bones are used in Facerig some calculations, like look-at-camera. Also the camera and head bones must have their respective Z axis pointing roughly one to another. That is the Z axis after Facerig import. It's possible that the handedness and axis conventions from the 3D software that generate the animation to differ from the same features of Facerig and thus the Z axis actually be Y or X in that original space. If

the model appears to load but is not visible on screen rotating the camera from the **original scene** (in your 3D application) might help finding the model. First, try pointing the camera along the worlds x,y,z, in both directions for each axis until you frame the model. Then if the model appears on the screen, check for other unwanted camera roll and pitch.
Note that for Camera bone the assumed FOV (Field Of View) is 30 degrees on the vertical axis. The FOV is **not** imported from the collada file, not any other camera specific property for that matter. To frame your avatar correctly, use vertical FOV of 30 and 1.77 aspect ratio, because this settings will be use in Facerig.

In order to have a proper scale and avatar orientation please import the **our example avatar** in your 3D application and use it as a template. Begin testing the data only with the geometry and idle files in order to establish proper basic setup.

The bones name “props”+something (like “propHead”) represents bones meant for accessories attachment. This bones are automatically added to prop bones list and can be used in Facerig to place other models on the avatar.

Pseudophysics driven bones
It's possible to have bones driven by springs with a “collision plane”. In the current version these bones act like a **pendulum** (like the ones driving avatars hair strands, Greta's hat or Alya's boobs), or a **reverse pendulum** (see “martian antenna” prop).

The set-up is like this: the physics bone is **directly under** or **directly above** a designated parent. It doesn't matter how the bones in this pair are called but you must avoid name clashing with the mandatory bones, listed above. There should be a third bone named like the parent of the physics bone plus the string “_planeNormal”. For instance if the parent is called “BipPhysBoneParent” this third bone should be “BipPhysBoneParent_planeNormal”. The Y axis of this third bone represents the normal of a collision plane, which can be used to restrict the physics bone movement. For instance if the physics bone deforms a hair strand and the artist wants to avoid this strand to penetrate the face, the collision plane should be defined as tangent to that particular face area, with it's normal (represented by that Y axis) pointing outward. Beware that in some 3D applications Y and Z axis must be swapped from what Facerig is using.

In such cases consider the Z axis for representing collision plane normal.
If the collision plane is not needed it's normal should point to the physics bone. Also note that even though the collison plane normal si defined by a third bone, it's origin is defined by the parent of the physics bone. All transformations are taking place in parent space.

In short, for each physics bone other two additional bones are needed, a parent and a collision plane stand-in. The physics bone is the one actually moving so you should skin the mesh to this bone or to something linked to this bone

Animations and expression poses

The animations must be provided in collada format.

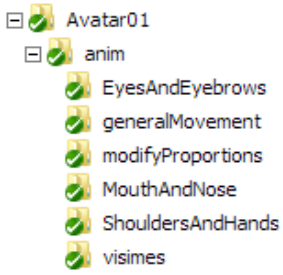
The animations must be at 30 frames per second.
The default scale factor should be set to identity, however is expressed in your 3D software (1,1,1 or 100,100,100). Scale factor can be animated.
The avatar model should face the same direction as the Z axis (in a Y up environment. If your system is Z up then the model should face Y axis)

Facering uses **base** and **additive** **animations**. Base animations, which are very few, like idle1, MouthOpen_base and MouthTongueBase, give **full transformations** to the bones. Additive animations only bring offsets relative to the transformation set by base animations. They only make sense if added (actually multiplied) with their base animation.

Additive animations should have non-moving bones in pose set by their base animation. For instance if you are making the frown animation for the left eyebrow, named “LeftEyebrow_D” then the only bones moving in that animation should be the middle and inner bones of the left eyebrow, all the rest should have the same **exact** transformation as in the first frame of their base animation, which is idle1. There are two reasons for that: first, most important, any offset in the additive animations bring a pose offset in Facerig. For exemple if an eyebrow animation brings an offset to neck, every time the avatar will move that eyebrow will also transform the neck. Second, if the bones move just when they suppose to, the script will identify the movement and only export animation for them, ignoring the rest, making debbug simpler and avoiding transformations made by lack of precision while adding to much transformation matrices.

In principle all additive animations have as base animation idle1 animation, more exactly they take reference the first frame of that animation. There are some exceptions to this rule:
- animations for open mouth which use as base “MouthOpen_base” animation, which in turn is added to idle1 animation. These animations are: “MouthOpen_pursedLips_LR”, “MouthOpenLeft_D”, “MouthOpenLeft_U”, “MouthOpenRight_D”, “MouthOpenRight_U”.
- animation for tongue while the tongue is sticking out. These are based on “MouthTongueBase” which is too added on idle1 animation. These animations are: “TongueOut_LR”, “TongueOut_UD”

Animation subfolder structure:



Needed animations (*frame 0 will start from idle position, unless stated otherwise*):

In **generalMovement** anim subfolder:

Avatar_FB: - The avatar leans forward from the waist at frame 0, goes through the idle pose at frame 15, then bends backwards at frame 30.

Avatar_LR: - The avatar leans left from the waist at frame 0, goes through the idle pose at frame 15, then bends right at frame 30.

Avatar_Twist: - The avatar twists left from the waist at frame 0, goes through the idle pose at frame 15, then twists right at frame 30.

Head_LR: - The avatar's head leans to the left at frame 0, goes through the idle pose at frame 15, then moves to the right at frame 30.

Head_Twist: - The avatar's head twists to the left at frame 0, goes through the idle pose at frame 15, then twists to the right at frame 30.

Head_UD: - The avatar's head looks up at frame 0, goes through the idle pose at frame 15, then looks down at frame 30.
idle1: - The base animation that most additive animations use as reference. This animation keeps the avatar in a neutral position, at **frame 0**. It can contain movement but this it should be very subtile as it will play over and over and could interfere with expressions. If idle contains movement that it should be loopable in order to avoid snapping at the end of the animation cycle.

In **EyesAndEyebrows** anim subfolder:

LeftEye_LR: - The left eye looks left at frame 0, goes through the idle pose at frame 15, then looks right at frame 30.

LeftEye_UD: - The left eye looks up at frame 0, goes through the idle pose at frame 15, then looks down at frame 30.

LeftEyebrow_D: - The inner half of the left eyebrow goes down, as in a frown expression.

LeftEyebrow_D_ext: - The outer half of the left eyebrow goes down.

LeftEyebrow_U: - The inner half of the left eyebrow goes up, as in a wonder expression.

LeftEyebrow_U_ext: - The outer half of the left eyebrow goes up.

LeftEyeClosed: - The upper left eyelid starts with a slightly more open position than on idle on frame 0, goes through the idle pose at frame 10 and becomes fully closed at frame 30.

LeftEyeSquint: - The lower left eyelid closes slightly at frame 30, leading to a squint.

LeftEyeWideOpen: - The left eye opens fully, moving the eyelids at the maximum distance apart.

RightEye_LR: - The right eye looks left at frame 0, goes through the idle pose at frame 15, then looks right at frame 30.

RightEye_UD: - The right eye looks up at frame 0, goes through the idle pose at frame 15, then looks down at frame 30.

RightEyebrow_D: - The inner half of the right eyebrow goes down, as in a frown expression.

RightEyebrow_D_ext: - The outer half of the right eyebrow goes down.

RightEyebrow_U: - The inner half of the right eyebrow goes up, as in a wonder expression.

RightEyebrow_U_ext: - The outer half of the right eyebrow goes up.

RightEyeClosed: - The upper right eyelid starts with a slightly more open position than on idle at frame 0, goes through the idle pose at frame 10 and becomes fully closed at frame 30.

RightEyeSquint: - The lower left eyelid closes slightly at frame 30, leading to a squint.

RightEyeWideOpen: - The right eye opens fully at frame 30, moving the eyelids at the maximum distance apart.

In **MouthAndNose** anim subfolder:

CheekPuff_L: - The left cheek inflates.

CheekPuff_R: - The right cheek inflates.

Mouth_pursedLips_LR: - The lips maintain a pursed position throughout the animation. At frame 0, they turn left, at frame 15 they go in a centered position, at frame 30 they turn right.

Mouth_unveilledTeeth_D: - The lower lip moves down, revealing the bottom teeth.

Mouth_unveilledTeeth_U: - The upper lip moves up, revealing the upper teeth.

MouthClosedLeft_D: - The closed left mouth corner moves down, as in a sad expression.

MouthClosedLeft_U: - The closed left mouth corner moves up, opening the lips and revealing the teeth, as in a wide smile expression.

MouthClosedLeft_U_visime: - The closed left mouth corner moves up while keeping the lips close together, leading to a less pronounced smile. It's called “visime” because is used along with visimes, alowing the user to have smile while speaking.

MouthClosedRight_D: - The closed mouth corner moves down, as in a sad expression.

MouthClosedRight_U: - The closed right mouth corner moves up, opening the lips and revealing the teeth, as in a wide smile expression.

MouthClosedRight_U_visime: - The closed left mouth corner moves up while keeping the lips close together, leading to a less pronounced smile. It's called “visime” because is used along with visimes, alowing the user to have smile while speaking.

MouthOpen: - The mouth starts from the idle position at frame 0 and becomes fully open at frame 30.

MouthOpen_base: - The base animation for all “MouthOpen” additive animations. This animation maintains the same fully open position throughout (frame 0 and frame 30). It's the same pose as in the last frame of the “MouthOpen” animation.

MouthOpen_pursedLips_LR: - The lips maintain a pursed position throughout the whole animation. It should be edited starting from an open mouth pose. At frame 0, they turn left, at frame 15 they go in a centered position, at frame 30 they turn right.

MouthOpenLeft_D: - Starts in the open mouth pose, left mouth corner moves down at frame 30.

MouthOpenLeft_U: - Starts in the open mouth pose, left mouth corner moves up, leading to a wide smile expression.

MouthOpenRight_D: - Starts in the open mouth pose, the open right mouth corner moves down.
MouthOpenRight_U: - Starts in the open mouth pose, the open left mouth corner moves up, leading to a wide smile expression.

MouthTongueBase: - The mouth starts from the idle position, and opens just enough for the tongue to stick out. It's last frame, with the tongue out, is the base pose for tongue animations, with the exception of “TongueIdle”.

NoseWrinker_D: - The avatars moves their nostrils slightly down.

NoseWrinker_U: - The avatars moves their nostrils slightly up.

TongueIdle: - The idle animation for the tongue consists of subtle movements that makes it look more natural than in a rigid position. It starts and ends in idle pose.

TongueOut_LR: - The tongue sticks out, it moves to the left at frame 0, goes to the MouthTongueBase last frame pose at frame 15 and moves to the right at frame 30.

TongueOut_UD: - The tongue sticks out, it moves up at frame 0, goes to the MouthTongueBase last frame pose at frame 15 and moves to the right at frame 30.

In **ShouldersAndHands** anim subfolder:

FingerL0_extFlex: - The left thumb goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerL1_extFlex: - The left index goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerL2_extFlex: - The left middle finger goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerL3_extFlex: - The left ring finger goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerL4_extFlex: - The left little finger goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerR0_extFlex: - The right thumb goes to extended (frame0) to flexed (frame 30) in a closed

position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerR1_extFlex: - The right index goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerR2_extFlex: - The right middle finger goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerR3_extFlex: - The right ring finger goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

FingerR4_extFlex: - The left little finger goes to extended (frame0) to flexed (frame 30) in a closed position, as when in a fist. In the middle of animation (frame15) it should be in idle pose.

HandL_closeDown_LR: - The left arm moves from left to right at a low height and close distance from the body (not fully stretched).

HandL_closeMiddle_LR: - The left arm moves from left to right at a medium height and close distance from the body (not fully stretched).

HandL_closeUp_LR: - The left arm moves from left to right at maximum height and close distance from the body (not fully stretched).

HandL_farDown_LR: - The left arm moves from left to right at a low height and maximum distance from the body (fully stretched).

HandL_farMiddle_LR: - The left arm moves from left to right at medium height and maximum distance from the body (fully stretched).

HandL_farUp_LR: - The left arm moves from left to right at maximum height and maximum distance from the body (fully stretched).

HandL_solo_LR: - The left hand moves left and right from the wrist at frames 0 and 30, while going through idle at frame 15.

HandL_solo_Twist: - The left hand twists left and right from the wrist at frames 0 and 30, while going through idle at frame 15.

HandL_solo_UD: - The left hand moves up and down from the wrist at frames 0 and 30, while going through idle at frame 15.

HandR_closeDown_LR: - The right arm moves from left to right at a low height and close distance from the body (not fully stretched).

HandR_closeMiddle_LR: - The right arm moves from left to right at a medium height and close distance from the body (not fully stretched).

HandR_closeUp_LR: - The right arm moves from left to right at maximum height and close distance from the body (not fully stretched).

HandR_farDown_LR: - The right arm moves from left to right at a low height and maximum distance from the body (fully stretched).

HandR_farMiddle_LR: - The right arm moves from left to right at medium height and maximum distance from the body (fully stretched).

HandR_farUp_LR: - The right arm moves from left to right at maximum height and maximum distance from the body (fully stretched).

HandR_solo_LR: - The right hand moves left and right from the wrist at frames 0 and 30, while going through idle at frame 15.

HandR_solo_Twist: - The right hand twists left and right from the wrist at frames 0 and 30, while going through idle at frame 15.

HandR_solo_UD: - The right hand moves up and down from the wrist at frames 0 and 30, while going through idle at frame 15.

In **visime** anim subfolder:

Visimes are animations used for lipsync, with each of their names representing poses accompanying the respective sounds. For example, visime_new_EH-AE has the jaw and lips positioned as when someone is making the EH or AE sounds. For each visime, frame 0 is identical with frame 30 and contains keys on the necessary bones in the required position for a certain visime. There no other keys in-between. These are the visime animations:

visime_new_AA
visime_new_AH
visime_new_AO
visime_new_AW-OW
visime_new_CH-J-SH
visime_new_EH-AE
visime_new_EY
visime_new_FV
visime_new_IH-AY
visime_new_L
visime_new_M-P-B
visime_new_N-NG-DH
visime_new_OY-UH-UW
visime_new_R-ER
visime_new_W
visime_new_X
visime_new_Y-IY

Directly in **anim** folder:

_frown: - Both eyebrows are in a frown position, which looks as when RightEyebrow_D and

LeftEyebrow_D are used at the same time. This animation is used as a reference for shading nodes activating specific normal maps.

_laugh: - Both mouth corners are up, leading to a laughing expressions. It looks as when MouthClosedLeft_U and MouthClosedRight_U are activated at the same time. This animation is used as a reference for shading nodes activating specific normal maps.

_unveilTeeth: - It looks as when both unveil teeth animations are activated at the same time. This animation is used as a reference for shading nodes activating specific normal maps.

_wonder: - Both eyebrowns are in a wonder position, which looks as when RightEyebrow_U and LeftEyebrow_U are activated at the same time. This animation is used as a reference for shading nodes activating specific normal maps.

cheekL: - It's an animation used for correcting transformations fo bones surrounding the mouth for models used retargeting tracking method. This surrounding bones are not tracked so they get transformations from this animations. They must have idle pose at frame zero, and the subsequant frame useally contains MouthOpen, Laugh, and purse lips poses. The animation is set on central and left side bones.
cheekR: - The same as cheekL animation only for the right side bones.

Avatar configuration file

The config file allow the user to set various parameters for its avatar.
If a .txt file is placed were the collada files are then is pickup by the script and copied as configuration file along with the rest of avatar's source data.

For the moment there is only one parameter exposed to user. Later on we'll expose more functionality to user.

set_head_axis axis
where axis can be any of these: x, -x, y, -y, z, -z. This represent the front pointing axis of the bone named “BipHead”. For instance if the axis pointing forward is y then the line should be:
set_head_axis y