Blu's Avatar Creation Standards

A guidline on good and optimal Avatar Creation in VRChat. This is all based on my own research and should be interpreted as a general idea of what to do when creating Avatars.

- Chaper 1: Guidelines
  - Introduction
  - Section 1: Model Rigging & Setup
  - Section 2: Textures
  - Section 3: Animator Controllers and You

Chaper 1: Guidelines

Introduction

Welcome! This is a guide geared sorely towards making the Avatars you create or buy as performant as they possibly can in VRChat.

VRChat for the longest time has been very flexible on PC in terms of how Avatars perform on the platform, but it has never been perfect. There are a lot of other things that can contribute towards how an Avatar performs on your Computer vs. how it is rendered to another user. Things such as Texture Memory (VRAM), Materials, and so forth. This guide will go over many of those things that should be taken as reference so as to make sure the Avatars you wear do not hinder another person's Computer.

> It is expected that you are already familiar with creating Avatars and using the VRChat SDK! If you are new to Avatar Creation, I highly suggest you follow some other guides before reading this! Everything in this book is going to go into a lot of technical detail that may be too much to comprehend for a first-time user, so I highly recommend familiarizing yourself with VRChat Content Creation before following this guide!

This guide of course is not perfect, but will provide a good sensible idea on things that you, the creator, should look out for when publishing your own Avatars to VRChat.

Section 1: Model Rigging & Setup

WIP!

Section 2: Textures

Every Avatar has a Texture that is used to present how it looks in it's final form. Without Textures, we'd be seeing flat-colored Objects. That would be so weird to see!

There are different types of Textures used on a Model. I will be going over the different types of Textures and how it affects your Avatar on a global scale.

2.1 - Image Format

Unity supports a wide range of Texture formats. When you upload your Avatar, they are re-encoded by Unity for use in-game at runtime.

It is recommended to keep your Textures in a lossless format so that the details are better preserved over the long term. The most common file format to use is PNG as it supports up to 16-bit Depth.

Of course, you are allowed to use other formats for your Textures if you so wish to. However, you must keep in mind of their properties as different formats can greatly affect it's lifetime in terms of quality and compression. Here are the following image formats that Unity supports:

- PNG (Recommended)
- TGA
- BMP
- DDS
- PSD
- JPG/JPEG

> **Never use JPG/JPEG for textures!** JPG is a *lossy* format, meaning that your source files are never kept at full quality. Use lossless formats like PNG, TGA, or PSD instead.

2.2 - Texture Resolution

Your texture resolution is the number of pixels in each direction. It is recommended that your Textures are Square and have dimensions that are a power of two - 256, 512, 1024, 2048, or 4096. Unity works best when textures are powers of two, especially when resizing textures.

You can author and store your textures at high resolution, and downscale them to a lower resolution with the texture import settings. More on this in Section 2.3.

Always use a power-of-two resolution! The most common power-of-two resolutions are 256, 512, 1024, 2048 and 4096.

If your Texture is set nowhere around a power-of-two, it can cause more blurry results when resizing to a power-of-two resolution.

2.3 - Your Textures and it's VRAM Consumption

One of the most misunderstood areas around using Textures on your Avatars is how much VRAM it can consume on your Graphics Card.

VRAM (Video Random Access Memory) usage is the amount of memory that will be used by a texture when it is loaded. All assets used by the GPU must be loaded into VRAM before they can be used, and there's only a limited amount of VRAM available on a single Graphics Card.

When the GPU runs out of VRAM, it will start to move textures into system memory, which causes latency and slows down the game. If you ever wondered "why is VRChat so damn laggy?" It's because of this exactly. Someone could be using an Avatar in your instance that is consuming a ridiculous amount of VRAM on your GPU! Whether it's intentional or not, it's always their fault that your game is running slow.

VRChat refers to it as "Texture Memory" as of the UI 2.0 Update, released in October 2022.

2.3.1 - Recommended Texture Sizes

Textures are the largest contributors to VRAM usage. Reducing the size and amount of textures will reduce VRAM usage and make your avatar more performant.

It is recommended that you keep your Textures anywhere around 2048 at best. If you prefer using 4096 resolution, use it wisely! Only use 4096 Textures in areas where you think you absolutely need it. If you use an anthropomorphic Avatar like a Nardoragon or a Rexouium, for example, it may be beneficial to prioritize it's Body texture to use 4096.

Avoid using Textures that are 8192 resolution! These Textures will consume the highest amount of VRAM on both yourself and everyone else showing your Avatar! So please avoid using 8192 Textures!

2.3.2 - Compression

By default, Unity will use Normal Quality on Textures imported. This means the details on your lossless Texture will be preserved well enough to be satisfactory. Sticking with Normal Quality is the most common setting used on Avatars.

This can of course be changed. You can either set it to Low Quality or High Quality. Both of which will have different results in terms of how the Texture will be rendered at runtime.

Be extremely careful if you choose to use High Quality or No Compression at all! While doing this can make your Textures look 1:1 with how it was originally, it will dramatically increase the Download Size of your Avatar and possibly your VRAM consumption!

Next to this is Crunch Compression. This is an algorithm that will (metaphorically speaking), squeeze your Textures down to a smaller size. The disadvantage of this is that you may see some loss of color in areas that have significant gradients, as well as introduced noise.

Crunch Compression your textures is in no way mandatory, however, it can greatly reduce the Download Size of your Avatar! This is especially helpful to users who have weaker internet. Be careful with doing this with high-res textures as they will be uncompressed at runtime when the Avatar is loaded for the user, causing a noticeable frame drop.

Crunch Compression does NOT help reduce VRAM! This is a common misconception so let's get this straight: Textures that are Crunch Compressed WILL get uncompressed at runtime once the Avatar is fully loaded for the user. The VRAM consumption will always remain the same as the resolution that the Avatar is set to!

SO DO NOT THINK that Crunch Compression will help your VRAM consumption, because it will never!

2.3.3 - Normal Maps

While the Main Texture you use will be the most frequent item you will choose, there are also other types of Textures that serve different purposes.

One of which is Normal Maps. These pink-like Textures are unique in their own way. With the right shading, Normal Maps can create an illusion that the Texture has a more detailed, physical appearance. Avatar Creators use Normal Maps to get away with expressing very nice details without having to sacrifice Polygons in order to get the intended look. Normal Maps are most commonly used on PBR-based Shaders since those Shaders are made to bring out more realistic detail.

Be careful with the resolution of your Normal Maps! Compared to a normal Albedo texture, Normal Maps can greatly contribute to higher VRAM consumption and file size, much more so than normal Textures!

Section 3: Animator Controllers and You

> This page is a WORK IN PROGRESS and it is not finished!

How does an Avatar come to life? What sorcery must be done to make things that look so simple actually work?

That my friend is the power of Unity Animators, the driving heartbeat of every Avatar that exists in VRChat! We use Animator Controllers to organize, sort, and determine how an Avatar behaves and looks at runtime. In this section, we'll be looking at the many different Layers that allows an Avatar to come to life!

3.1 - What does each Animator do?

By default, VRChat uses 5 Playable Layers. Each of them serve their own purpose. If your Avatar is set to use any of the defaults, then it will always play the default Animator Controller for that specific Layer.

In most cases, you will not need to make each one separately by yourself. Most likely, you'll only edit 2. I'll explain more what are most commonly modified further below. But either way, it is pretty important to get an understanding of what each of them does and how it will affect your Avatar.

> All example Animator Controllers and it's sister .anim files that VRChat uses are included in all versions of the VRChat SDK for Avatars 3.0. To find them in projects created using Creator Companion, they will all be located in: `Packages/VRChat SDK - Avatars/Samples/AV3 Demo Assets/Animation`.
>
> Note that `Packages/VRChat SDK - Avatars` folder may show as `com.vrchat.avatars` if viewing it's files through Windows Explorer.

3.1.1 - Base

Default Animator Controller: `vrc_AvatarV3LocomotionLayer.controller`

Commonly referred to as "Locomotion," this Layer is the sole heartbeat behind all kinds of Avatar Movement! This includes things such as Walking, Running, Falling, Crouching, Sitting, and Jumping!

If you ever wanted to know where all of that movement happens from, this is the Controller that is responsible for all of that!

In many cases, this layer is rarely edited by the end-user because this Layer is very complex! It contains a vast amount of Blendtrees, Transitions, and animation states that reference to Humanoid Animations. These Humanoid .anim files are far different than any kind of Animation! They specifically control the IK (aka Inverse-Kinematics) of a Humanoid Avatar Rig.

> **Due to the complexity and unique behavior of the Base controller, it is aimed towards Advanced Users only!** Do not add any of your own Controllers to Base unless you know exactly what you are doing!

> Be extremely careful when editing and/or adding Animations to Base! Any mistake you make could end up creating wild and unintentional Locomotion behavior with your Avatar! We recommend copying the default Animations so that the original files aren't affected.

> Humanoid Animations are VERY HARD TO EDIT on your own without the right plugins! You'll see why if you attempt to look at the keyframes of those .anim files.
>
> Use a plugin such as **Muscle Animation Editor** or **Very Animation** to edit them, as they are designed to create and/or edit Humanoid Animations. They can be found on the **Unity Asset Store**.

3.1.2 - Additive

Default Animator Controller: `None (User Dependent)`

Additive is a special layer. It is added on-top of the Base controller for animations that should "add-on" in addition to an existing Humanoid Bone affected by IK. This can be used for things such as a "breathing" animation that uses Bones to make that happen.

This Layer is rarely used in most cases.

> Animations in Additive SHOULD ONLY AFFECT Humanoid Bones that are driven by IK! If you have non-humanoid bones such as Ears or a Tail, use the Gesture Controller! More info on Gesture in Section 3.1.3.

3.1.3 - Gesture

Default Animator Controller: `vrc_AvatarV3HandsLayer.controller`

The Gesture Layer is used for Animations that require to act on individual body parts while still playing the underlying Animations for the rest of the body

This is an important layer. Gesture is used to determine how Fingers on an Avatar's Hand should animate depending on which Hand Gesture the user is attempting to pose. They would be either Fist, Open, Point, Peace, Rock N' Roll, Finger Gun, or Thumbs Up.

If you have an Avatar that will have Idle or Puppet Animations on non-Humanoid Bones such as Ears or a Tail, use the Gesture Layer! In addition, it is highly advised that any kind of non-Humanoid Bone that will have it's Transforms animated to go into the Gesture Layer!

The first 3 Layers are mainly important. They require the usage of **Masking** in order to direct the Animations to affect a certain part of your Avatar. For VRChat, the default Masks for these are located in `Packages/VRChat SDK - Avatars/Samples/AV3 Demo Assets/Animation/Masks`.

Here are the Masking Requirements to follow for the first 3 Layers of the Gesture Controller:

- AllParts: `vrc_HandsOnly`
- Left Hand: `vrc_Hand Left`
- Right Hand: `vrc_Hand Right`

> *To use Hand Gestures in VRChat using an **Oculus** or **Vive Controller**, please refer to the image chart shown on VRChat's Documentation here.*
>
> *Desktop Users will use Keyboard Shortcuts to use Hand Gestures in VRChat. Fo more information, see the Keyboard & Mouse bindings on VRChat's Documentation here.*

> When making your own Gestures, refer to the VRChat Documentation on when the `GestureLeft` and `GestureRight` Int Parameters are triggered here. These Parameters are already synced across the Network using the IK solver, so you only need to add these to your Gesture Controller. Do not add them to your Expression Parameters!

> **Valve Index Controllers** will most likely at times never play the finger animations as those are overridden by it's native Finger Tracking (obviously).

3.1.4 - Action

Default Animator Controller: `vrc_AvatarV3ActionLayer.controller`

The Action Layer is for Humanoid Animations that should override all other Layers, for times when you would want to play an "Animated Emote."

If you have a Dance Emote, this is where you would want to put it! Some other Animations can be found here as well, such as the AFK Animation (triggered by the `AFK` Bool).

> Before adding your own Animations to Action, make sure to use the `VRC Playable Layer Control` component! Since the Action Layer is Weighted to 0 by Default, you need to override it's Weight before an Animation is played!
>
> In addition, you will need to make use of the `VRC Animator Tracking Control` as well. This animation state component allows you to filter out any of your own movements before and after an Animation plays!

3.1.5 - FX

Default Animator Controller: `None (User Dependent)`

The FX is a special layer! What's so unique about this Layer is that **everything in the FX is copied over!** In other words, Animations that DO NOT affect Transforms whatsoever shall be put into this Layer!

The FX is the most common Layer to be edited by the end-user. This is where the majority of the Toggles and other magic that you add to your Avatar should be animated from!
Want to put an Animation that changes a Material? Put it in the FX!
What about Object Toggles? Yes! Put it in the FX!
Blendshape Animations? Absolutely!
Wanna enter the battlefield with that cool Lightsaber you got for Christmas that can be thrown like a true Jedi? This is where you put those Animations in!

To put it simply, this is the Layer that your Object Toggles and other magical things that would impress Gandalf should go to! Many of the magical things that users show off to "wow" the whole audience were primarily Animated in the FX! If you ever watched a Tutorial somewhere on how to make a simple object Toggle, it is super often to see the FX being used to create those Toggles!

On some Avatars, you may also see the first few Layers to be labeled similarly to how Gesture labels them. Things like `Left Hand`, `Right Hand`, or `Both Hands`. **The Animations used on these are NOT like the ones seen in Gesture!** In the FX, they are used for Facial Expressions that are driven by Blendshapes on a Skinned Mesh, for example. They also never use Masks in the FX as they are driven entirely by the `GestureLeft` and `GestureRight` Int Parameters. So if you want to have your Avatar have Facial Emotions triggered by your Hand Gestures, this is where you should animate them!

> Try to avoid transitioning from "Any State" for your Facial Expressions driven by your Hand Gestures and instead use a transition pattern such as `Entry -> Idle -> [your Expression`

`here] -> Exit`, for example. This is because any Condition that is from an "Any State" is checked for every frame, which can be expensive to your CPU compared to a normal transition pattern.

DO NOT USE MASKS ANYWHERE IN THE FX! As of the latest VRChat SDK versions, any Mask used in the FX will greatly conflict (and possibly break) any Finger Animation used with the Gesture Controller! **This is especially important to know as it mainly affects users with Oculus or Vive Controllers!**