

Aavegotchi Unity SDK - Readme

Introduction [↗](#)

The Aavegotchi Unity SDK is meant to make developing games with the Aavegotchi IP in 3D and Unity as easy as possible.

Content Overview [↗](#)

- The basic (naked) 3D model of an Aavegotchi
 - Support for all collateral and eye shapes
 - All 3D assets for wearables
 - An animation library to bring the aavegotchi to life
- 3D Lickquidator assets (meshes / textures / animations) (originally created for Gotchi Guardians)
 - Dire Tire
 - Funguy
 - Steady Lad
 - Licker
- A sample scene known as the Aavegotchi Dressing room.
- Dependencies:
 - Addressables (used to load wearables)
 - Text Mesh Pro (used in the dressing room)
 - DOTween (manually added to the package because DOTween is awesome)

No Wallet Connection [↗](#)

Currently we are not providing any login or fetching of wallet data. We may add this in the future but it is NOT necessary to get started.

Anatomy of an Aavegotchi [↗](#)

Aavegotchis are quite complex creatures! The SDK is built to handle as much of this complexity for you as possible.

To learn more about Aavegotchi traits, you can find documentation here: <https://wiki.aavegotchi.com/en/traits>

i For example, some hand wearables like the energy gun can override a body wearables sleeves! Don't worry, this is all handled automagically for you 😊

Aavegotchi_Data [↗](#)

This simple data class represents the configuration of an Aavegotchi. This can be mapped to an aavegotchi config from a players wallet to bring an aavegotchi to life. It can also be used to create any combination possible to create things like NPC characters.

Table of Contents

Introduction
Content Overview
No Wallet Connection
Anatomy of an Aavegotchi
Aavegotchi_Data
Aavegotchi_Base
Wearables
Animations
Idle Animations
Moving
Actions
States
Misc
Lickquidator Assets
Supported animations:

Current Data	
Haunt ID	0
Collateral Type	Eth
Eye Shape	Common 1
Eye Color	Common
Body_Wearable ID	0
Face_Wearable ID	0
Eyes_Wearable ID	0
Head_Wearable ID	0
Pet_Wearable ID	0
Hand Left_Wearable ID	0
Hand Right_Wearable ID	0
Skin ID	0

⚠ **Haunt ID** is something defined in a really aavegotchi but is not actually used in this SDK.

As of writing of this documentation, there were two Haunts, each having their own collateral types and mythical eye shapes.

Instead of adding more complexity to this data, we simply allow all eye shapes and collateral types regardless of haunt ID.

⚠ **Skin ID** is not used directly by the SDK. It is used by Gotchi Guardians to support Guardian skins like the Ghost Pirate

You should generally leave this at 0. Any other value will effect wearable loading.



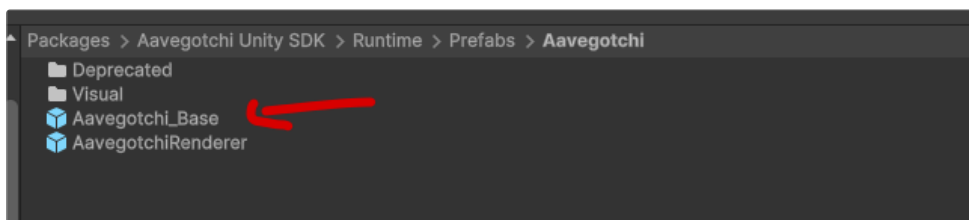
It does this by looking for a new wearable package (for all equipped wearables) with the skin id appended to the wearable file name.

For example, lets say I have a skin ID of 10, and a hat wearable of ID 1. Normally it would load wearable_1, but since a skin ID was provided, it will try and load wearable_1_10.

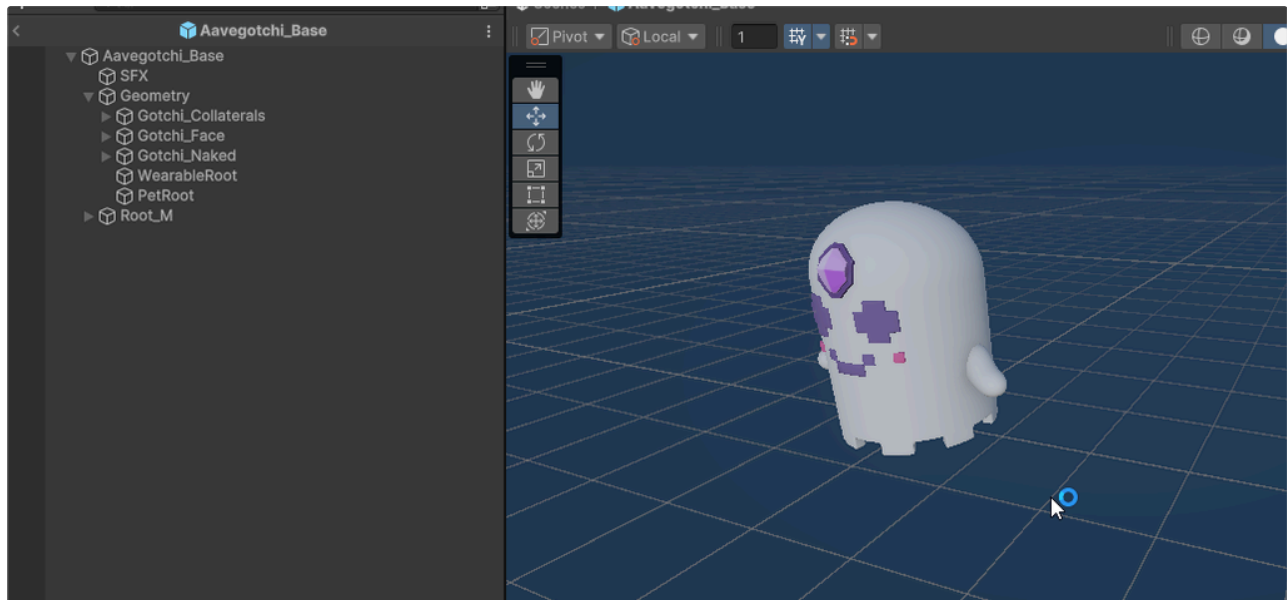
There currently is no mechanism to handle different Skin IDs per wearable. Its all or nothing.

Aavegotchi_Base [↗](#)

This code / prefab is one you can easily build with and can be found here:



Since many different games / projects might want to implement different ways of controlling their Aavegotchi, the SDK does **NOT** provide **ANY** such mechanisms. For example, Gotchi Guardians created its own prefab that contains an Aavegotchi_Base that it controls.



The complexity under the hood here is enough that it is **STRONGLY RECOMMENDED** to NOT play with this prefab if you can avoid it.

By default it contains all the scripts and game objects related to the functioning of a naked Aavegotchi.


It also handles ALL wearable loading.

To change the configuration of your Aavegotchi, simply call **UpdateForData** with your desired **Aavegotchi_Data** and it will take care of all the rest for you.

The currently loaded data is exposed in the inspector if you wish to modify it. However, for performance reasons, you must check the refresh checkbox for your changes to take effect. This is not necessary when calling **UpdateForData** with code.

Wearables

You can also find the full list of all wearables and their ID's here: <https://wiki.aavegotchi.com/en/wearables>

 Please note that the SDK does NOT guarantee that correct wearable ID's are applied to a specific wearable slot.

For example, the Steampunk goggles is an eye wearable. The SDK does NOT validate this and behavior is unpredictable if you assign its wearable id (199) to the wrong slot (like body).

Most wearables (except most hand wearables) are simple skinned mesh renderers that are loaded from addressables.

Because skinned mesh renderers are waiting for have their bone data added, they will not render in the prefab until they are instantiated and hooked up by the SDK.

Hand wearables are broken into 4 categories:

- Melee
- Ranged
- Shield
- Grenade

Depending on the hand type assigned, it will attach to a different bone root that is animated accordingly.

Generally this is not something you need worry about BUT enables us to have a smarter animation system.

Animations [↗](#)

Animations provided for the aavegotchi can be broken into the following major categories:

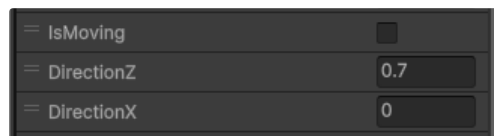
- Idle
- Moving
- Actions
- States

Idle Animations [↗](#)

This is the default state of an aavegotchi. Random values passed to animation power some idle breaking animations

Moving [↗](#)

Moving is a bit trickier and your needs might vary depending on how you plan to move you aavegotchi. It is controlled by 3 different animator variables:



i Remember, if your scripts have a reference to the animator of **Aavegotchi_Base** is controlled by calling the correct methods (depending on the type of variable).

```
currentPosition = currentPosition;  
Animator.SetBool("IsMoving", true);  
}
```

While it is up to your code to decide how to rotate or move the character, these parameters allow you to animate the character moving forward / backwards and straffing left / right (or any logical combination thereof).

Direction Z controls whether the Aavegotchi leans forward (+1.0) or back (-1.0)

Direction X controls whether the Aavegotchi leans left (-1.0) or right (1.0)

Values in between effect the strength of the leaning.

Actions [↗](#)

These are quick animations and are designed to override whatever the character is doing and play immediately. Thus these are controlled by triggers

i These are activated by code thusly:

```
Animator.SetTrigger(triggerName);
```

The list of possible actions are:

- Defeat
- Victory
- Victory_Fast
- Ability_Spooky
- Ability_SpinAttack
- Ability_RushAttack_Unleash

- Block
- Attack
- Jump
- ThrowGrenade
- Shoot
- Spawn
- Action_Trigger
- Action_Clapping
- Action_Dance1
- Action_Dance2
- Action_FollowMe
- Action_Refuse
- Action_Salute
- Action_Waving

States

States are changes to the aavegotchi that continue until they are turned off. They often override other states like idle or Moving.


Actions will take over the aavegotchi but will return to the correct state after completing. States are not designed to compete with each other (except Idle / Moving) and behavior is normally undefined if you turn on multiple conflicting states.

The **Dead** state is an exception as it is all consuming and even disables most actions from overriding it.

The states supported at the time of writing this are:

- Jumping
- Dead
- Daashing
- GrenadeAiming
- Aiming
- Building
- Blocking
- Rushing
- Action_Dance1_Loop
- Action_Dance2_Loop

Any overlap between the States and Actions are often due to the different demands of FPS gameplay like **Spirit Force Arena** or more RTS like gameplay of **Gotchi Guardians**.

 For example, **Gotchi Guardians** wants to just play a throw grenade animation and be done with it (Action)

However **Spirit Force Arena** wants to play a grenade wind up animation until the player has decided where to throw it (State for wind up + Action when ready to throw).

The SDK allows both options to work in harmony.

Misc

There are a few parameters that you should avoid manually editing:

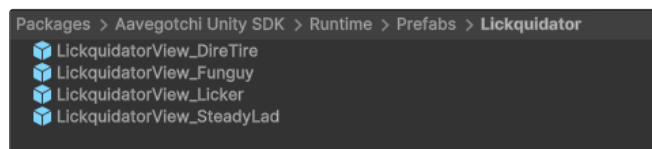
RandomSeed	0
MeleeHands	0
GrenadeHands	0
ShieldHands	0
RangedHands	0

These are all controlled by the SDK and handle things like animation variants or help reflect what wearables are equipped in the aavegotchis hands.

For example if you equip a gun in the left hand and a sword in the right, your aavegotchi will always melee with his right hand and shoot with his left. When the preferred item type

Lickquidator Assets [↗](#)

Provided lickquidator assets can be found here:



These are much simpler and it is recommended to build atop them similarly to how we described building atop the Aavegotchi. They contain the meshes, materials, textures and animation data all ready for you to use.

Supported animations: [↗](#)

Generally these support simple animations like:

- Idle
- Move
- Attack
- Death (Some deaths change depending on whether you marked it as moving or not)

If you see a RandomSeed Parameter in the animator, you must seed it with a random value between 0.0 and 1.0 at regular intervals (can be every frame) so that idle breakers and variants work correctly.