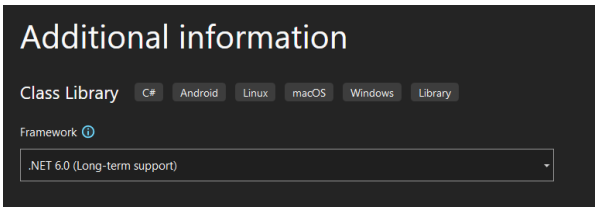
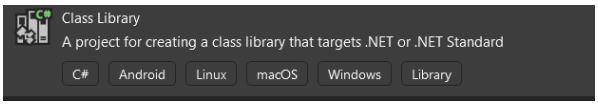
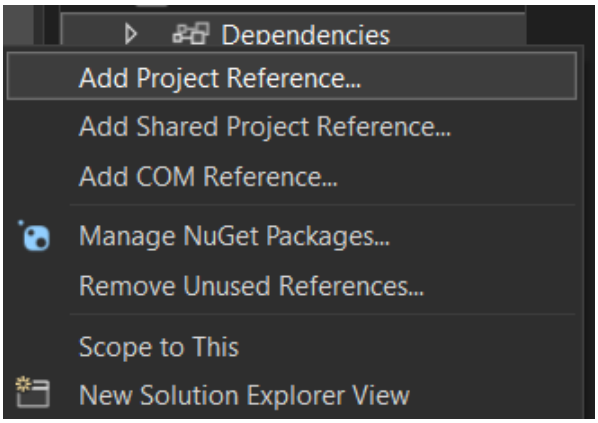
Food Modding Guide

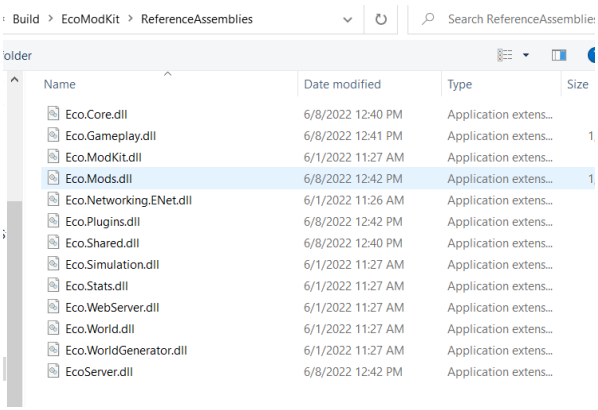
Server

Create a class library project in the latest version of Visual Studio with .Net 6.0 as the target framework. Then right-click dependencies and add project reference, on it browse for the download reference assemblies dlls.

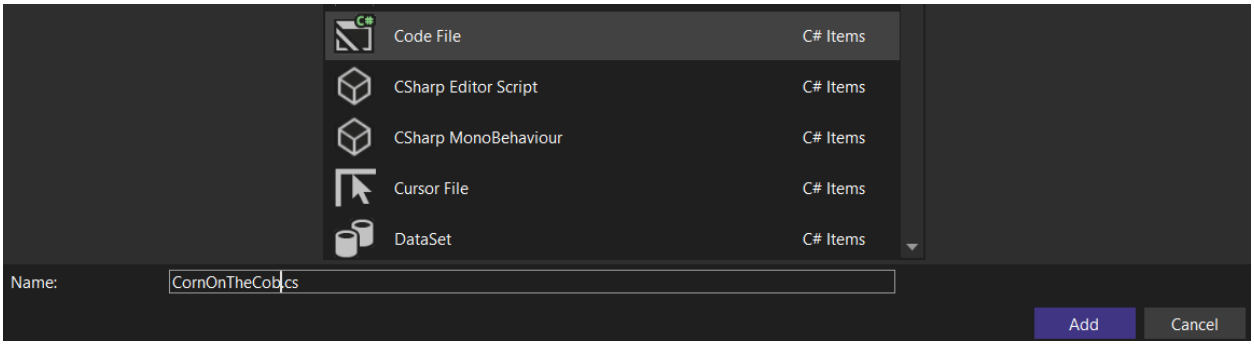








Right click project then add a new Item, add a new code file, rename it appropiately



On the Code file use the following namespaces that should cover all of the definitions used on a new Food Item.

```
using System;
using System.Collections.Generic;
using System.Linq;
using Eco.Core.Items;
using Eco.Gameplay.Components;
using Eco.Gameplay.Items;
using Eco.Gameplay.Players;
using Eco.Gameplay.Skills;
using Eco.Shared.Localization;
using Eco.Shared.Serialization;
```

After that create the CornOnTheCobItem partial class inside the namespace Eco.Mods.TechTree, make sure that it is named appropriately by having the suffix “Item”. Also that it inherits from FoodItem class. Implement abstract members of **Nutrition**, **Calories** and **ShelfLife** and add [Serialized],[LocDisplayName()] attributes to the class as shown in the image.

```
namespace Eco.Mods.TechTree
{
    [Serialized]
    [LocDisplayName("Corn on the cob")]
    public partial class CornOnTheCobItem : FoodItem
    {
        public override Nutrients Nutrition => new Nutrients { Carbs = 12, Fat = 2, Protein = 3, Vitamins = 11 };
        public override float Calories => 250;
        public override int ShelfLife => 86000;
    }

}
```

After adding the necessary stuff we can add the [Weight] and [Tag] attributes to the class, which are not necessarily required but rather recommended. Also, override **DisplayNamePlural** and **DisplayDescription** so that the item can display in the interface a name and a description, like so.

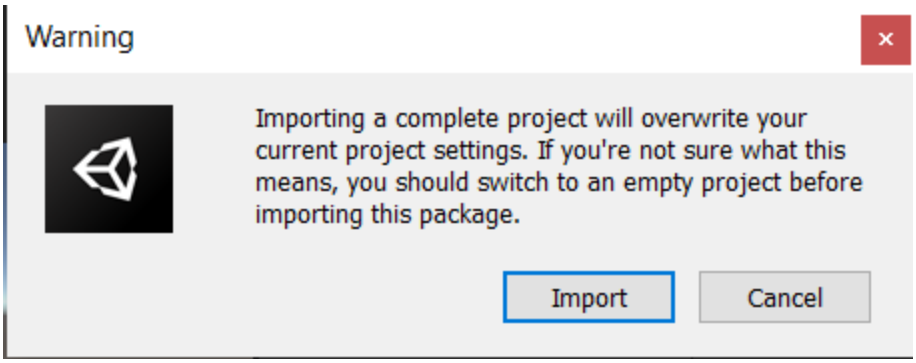
```
namespace Eco.Mods.TechTree
{
    [Serialized]
    [LocDisplayName("Corn on the cob")]
    [Weight(300)]
    [Tag("BakedVegetable", 1)]
    [Tag("BakedFood", 1)]
    public partial class CornOnTheCobItem : FoodItem
    {
        public override LocString DisplayNamePlural => Localizer.DoStr("Corn on the Cob ");
        public override LocString DisplayDescription => Localizer.DoStr("A warmly colored kernel studded vegetable.");

        public override Nutrients Nutrition => new Nutrients { Carbs = 12, Fat = 2, Protein = 3, Vitamins = 11 };
        public override float Calories => 250;
        public override int ShelfLife => 86000;
    }

}
```

Client

First things first Import the modkit into your unity project.



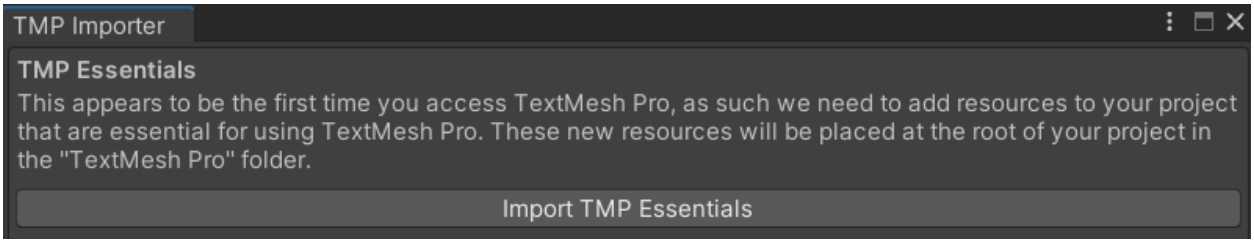
Import the assets you're going to use. In this case, I'm importing the corn mesh divided into parts, the FPV hands necessary animations, and the corn Icon, I'm not importing the TPV hands animations to show how you can use animations already found within the game. Either way, you can create your own hand animations by using the .FBX hands model found in **Assets/EcoModKit/Assets/Food/HandsModel/Avatar@FPV_Hands_Tpose.FBX** in 3d animation software such as blender.

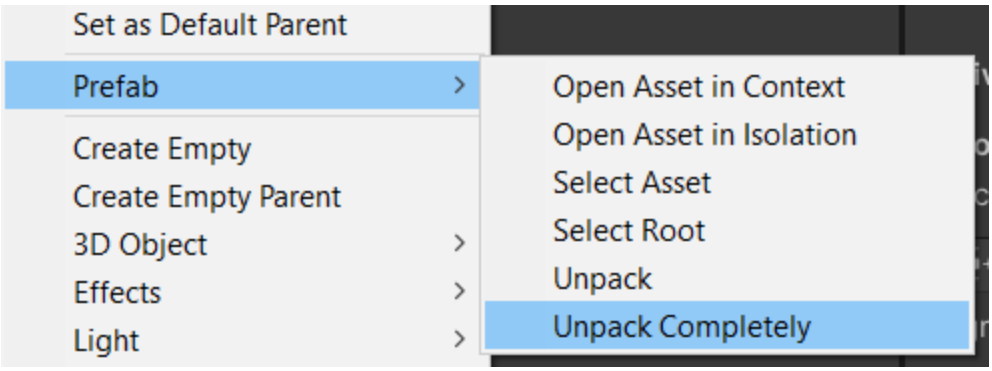
Finally, delete all the objects by default in the sample scene.

Base object setup

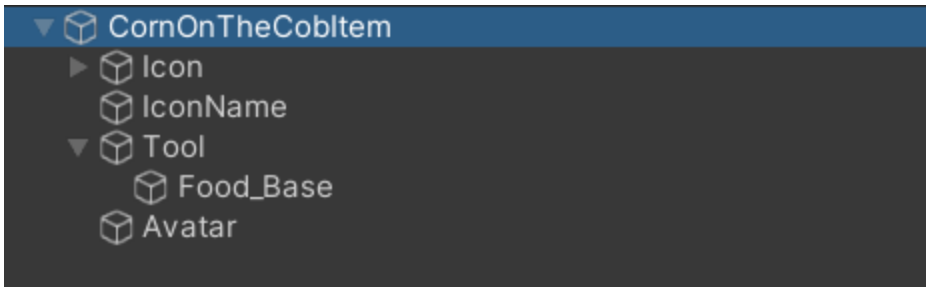
We want to first build the base of our food object, from which we can use as the TPV version and then override it and create the FPV version.

Import FoodItemTemplate into the scene, Import TMP essentials if a popup appears, then after getting to **FoodItemTemplate** prefab on the scene we're going to Unpack it completely and rename it correctly, in this case, **CornOnTheCobItem(**the same as the partial class in Server side of things that represents the item**)**, it is important to keep the naming convention to end in "Item"

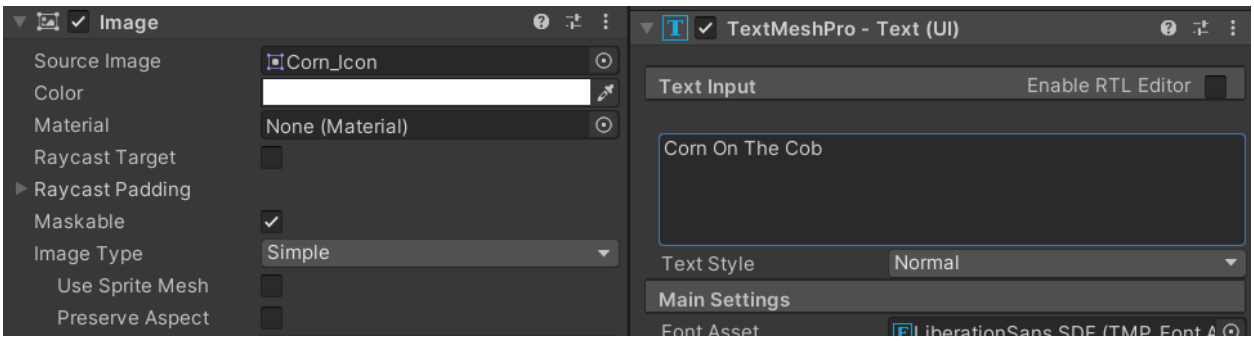




This is the structure we're left with after importing the prefab into the scene , unpacking it and renaming it.



Change the sprite on the **Image** Component in both the **Foreground** object and **FullImage** object, as well as write the correct name on the **TextMeshPro** component in the **IconName** object



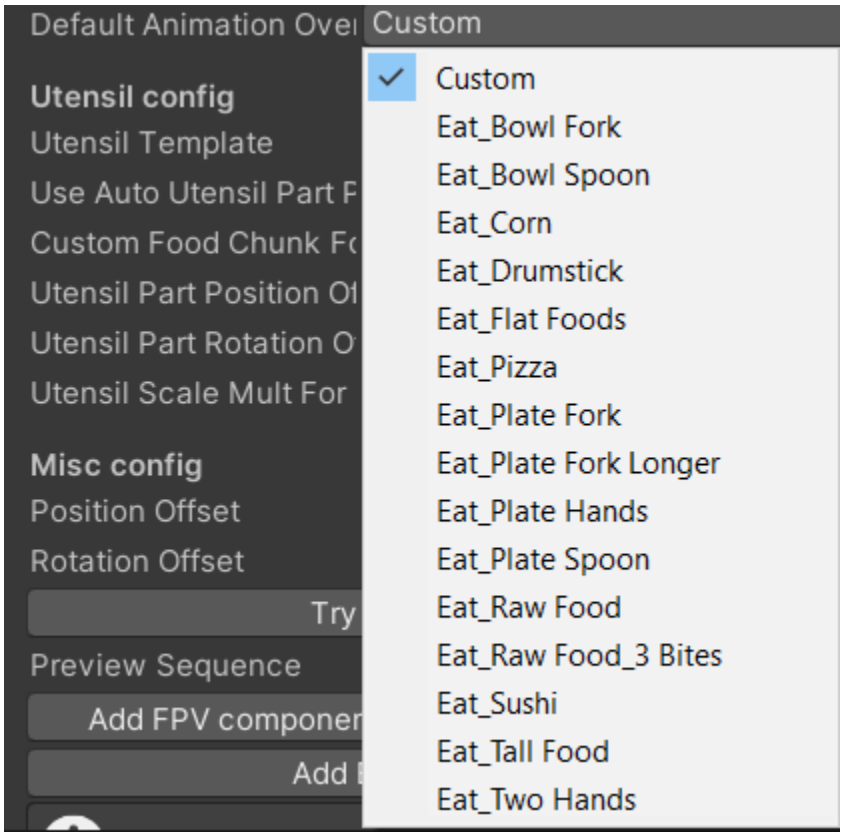
Components Setup

Now let's focus on the tool object, specifically the Food_FPV object, there we will find the following components: **BiteableFoodObject**, **ToolInteraction**, and **Food Effects.**

Biteable Food Object

Property	Description
Food Parts *	Reference to all the GameObjects that represent all the different bites that it takes to consume. E.G the different meshes of the corn.
Bite Sequence *	Refers to how the GameObjects that compose meshes will behave with each bite, you could have a tomato for example which is just various types of the same mesh in different states or a salad which is composed of various parts. There are already 2 options defined for you to choose on **Assets/EcoModKit/Assets/Food**, or alternatively, create a new one inheriting from **FoodBiteSequence** class.
Plate Template	It refers as its name implies to the object in your food that sort of acts like a container, E.G: a **bowl** in a salad.
Default Animation Override	It refers to the hand's animations to be used when eating, you can either create a **Custom** one using **Custom Animsets Override** or choose one of the already predefined animations that we already have in ECO.
Utensil Config	
Utensil Template	refers to the model used as a utensil like a fork or a spoon
Use Auto Utensil Part Placement	
Custom Food Chunk for utensils	refers to the model used when grabbing pieces of chunk from the food
Utensil Part Position Offset	Helper to better adjust the utensil position within the hands model.
Utensil Part Rotation Offset	Helper to better adjust the utensil rotation within the hands model.
Utensil Scale	
Misc Config	
Position Offset	Helper to better adjust the food position within the hands model .
Rotation Offset	Helper to better adjust the food rotation within the hands model .

- **Default Animation Override options**



Lastly, there are tools to help you set up the object as well as visualize it. **Add FPV components** will add **Tool Interaction**, **Food Effects** and **Perform Interaction. While Add TPV components** will only add **Tool Interaction** and **Food Effects.**

Food Effects

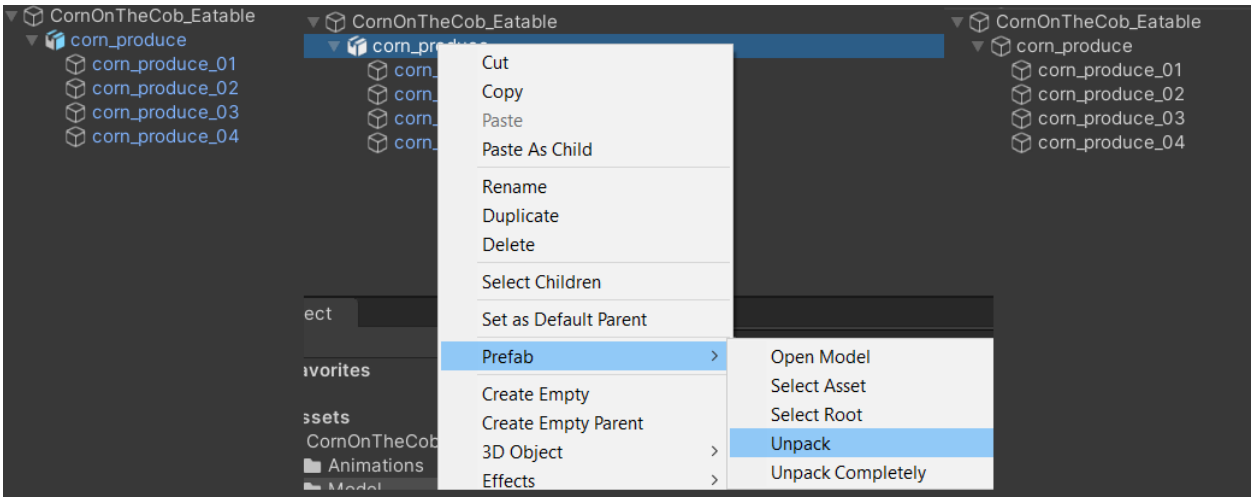
Sequence Interaction Effects	Should the interaction effects play all at once or in a sequence with each bite?
Interaction Particle Systems	Interaction Particles are for when the player interacts with the food before taking a bite. eg. spoon entering the soup
Sequence Bite Effects	Should the bite effects play all at once or in a sequence with each bite?

Bite Effect Particle Systems	Bite Particles are for when the player actually takes a bite.
Food Sound Type	audio category for the chewing sounds on each bite
Interaction Sound Type	Sound to play when the player interacts with the food eg. Spoon entering soup
Mouth Effect Transform	Optional: If sequencing mouth effects then you can parent the individual effects to this transformation and it will be positioned to the mouth instead of the parent effect
Mouth Offset	When passing an **InteractionVFX** through the Animation event(see animation events down below), the bite effects are moved to the camera's position plus this offset in order to simulate mouth position
Mouth Spray Random Angle	How much variety is there in the angle of juicy mouth effects

Tool Interaction

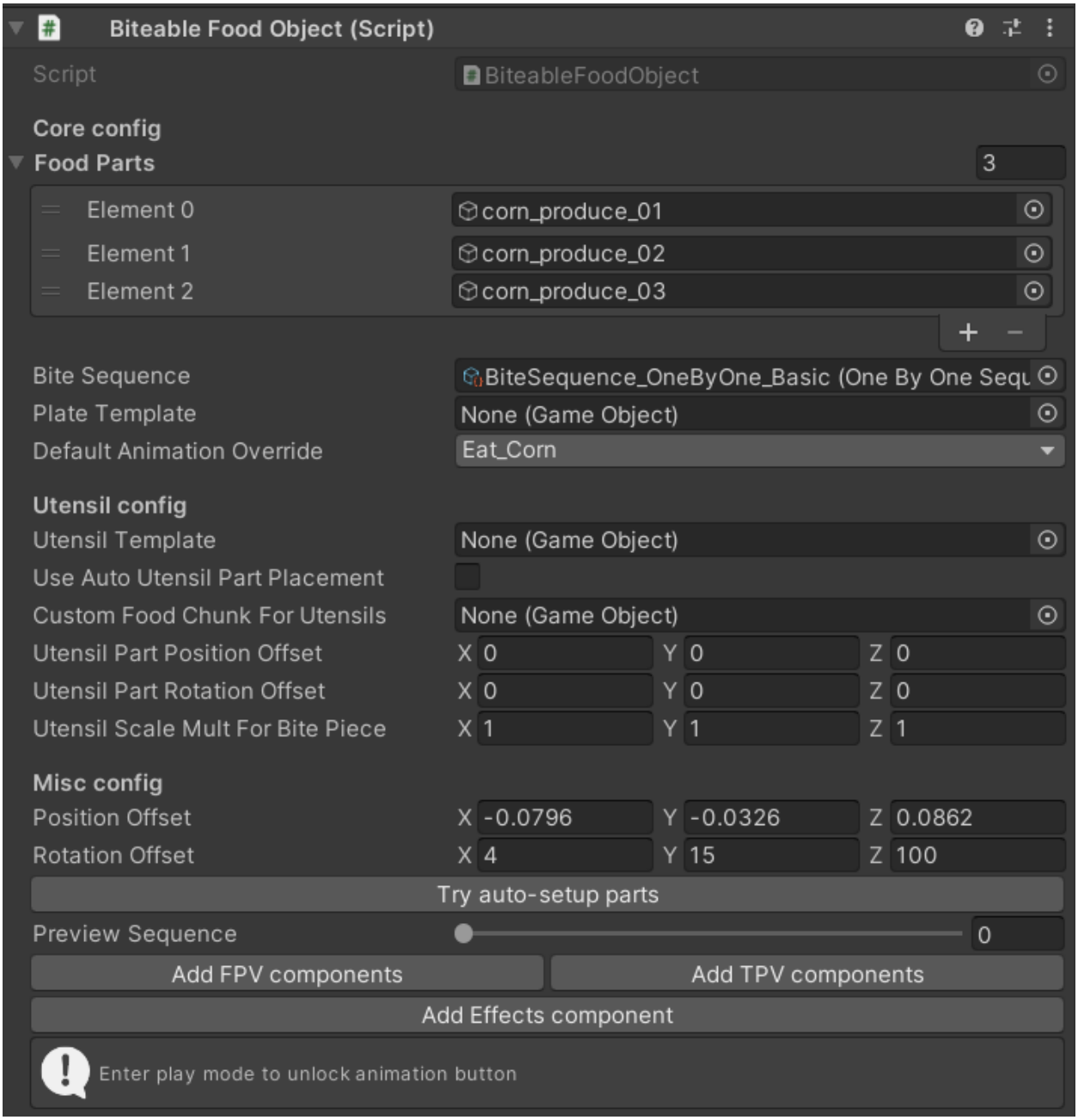
Custom Animset Override	Assets that allows for the FPV or TPV hands animations to be overridden by custom ones.

Continuing making the corn on the cob mod…, First I'll rename Food_Base object into something more appropriate like **CornOnTheCob_Eatable** and import the food parts mesh as a child of this object. Remember that this will constitute the base for our food object.



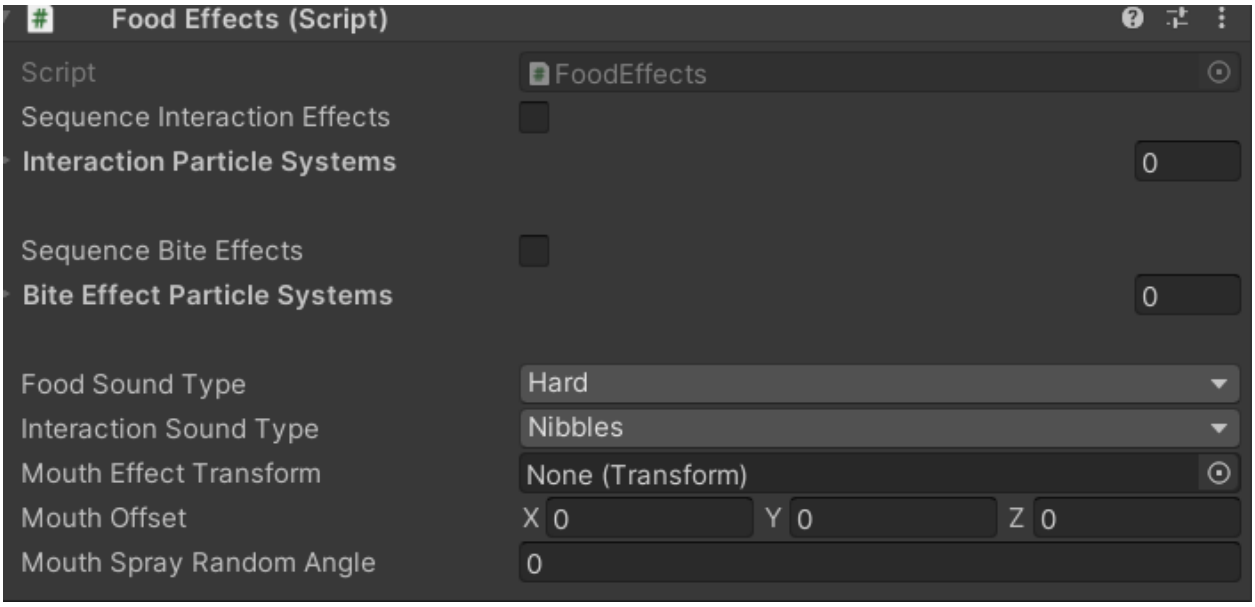
- After importing the mesh into the scene you can add the parts automatically by pressing the **TryAutoSetupParts** button, in this case, I removed the last part since that is the stem of the corn.
- I Set Bite Sequence to OneByOne since the corn mesh will disappear piece by piece after each bite.
- Default animation override is set to Eat_Corn, those animations are already defined in Eco.
- Lastly some adjustments to the object's position and it's offsetsin order to fit the corn correctly in the hands of the player while it plays the animations.

You can see the final result of how I ended setting up the BiteableFoodObject.

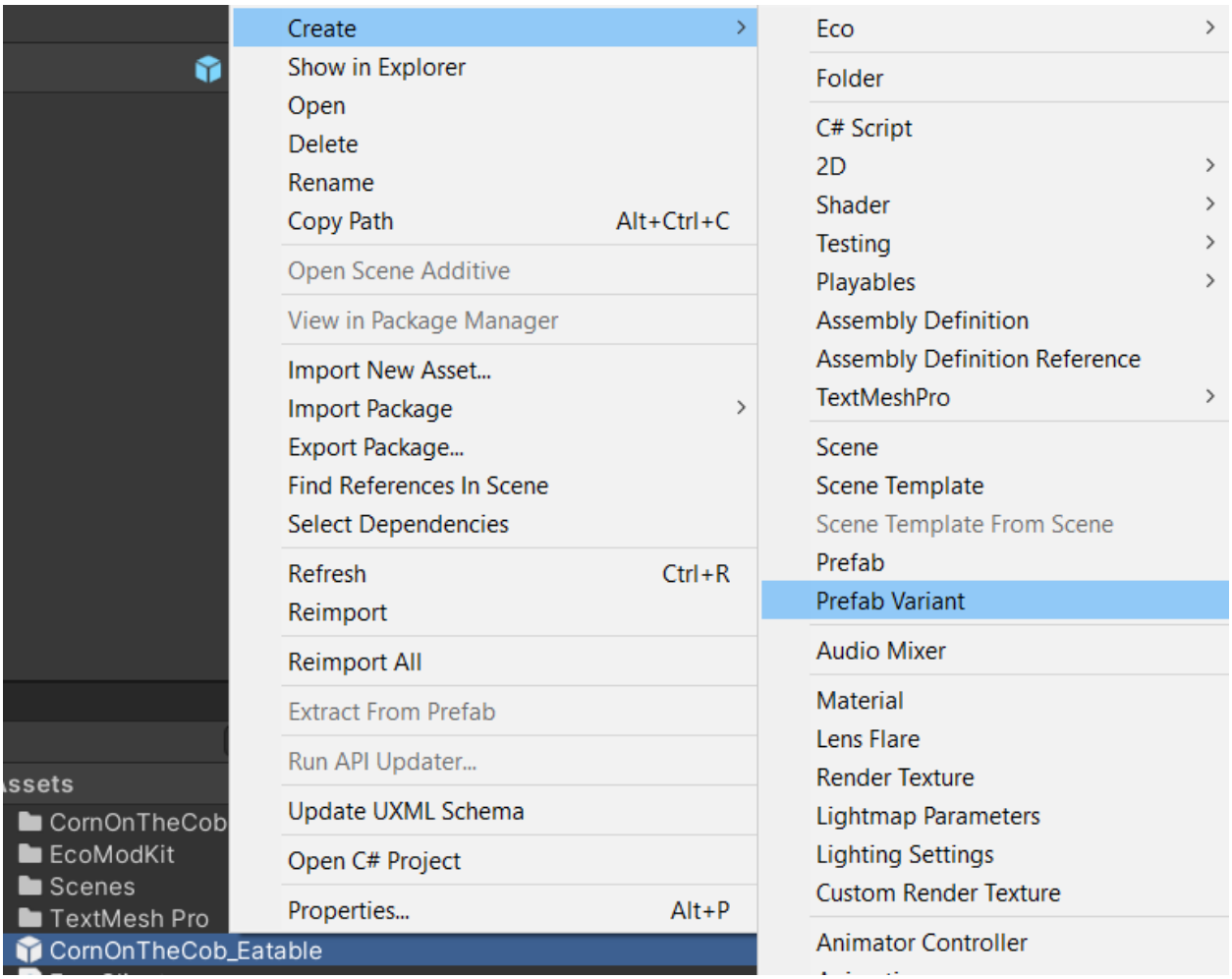


Regarding **ToolInteraction,** I left it as it is for now.

On Food Effects I changed the **Food Sound Type** to Hard and the Interaction Sound Type to Nibbles.

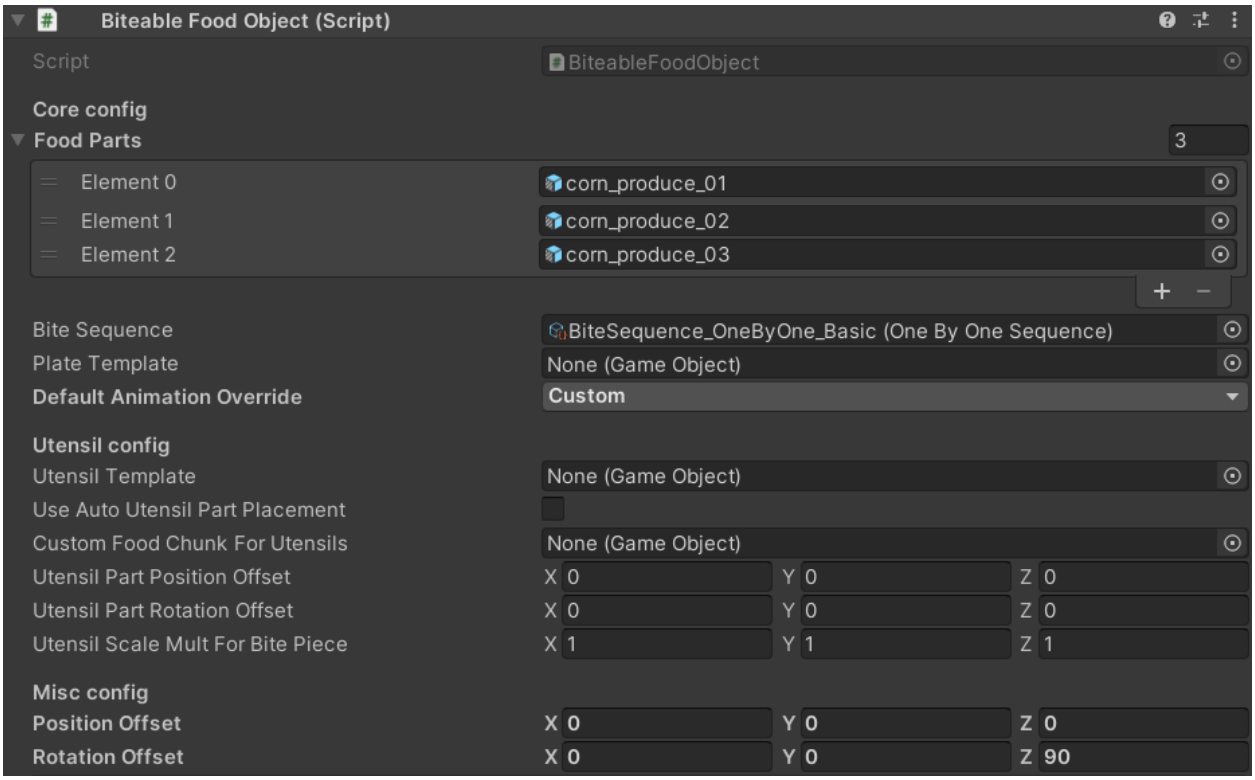


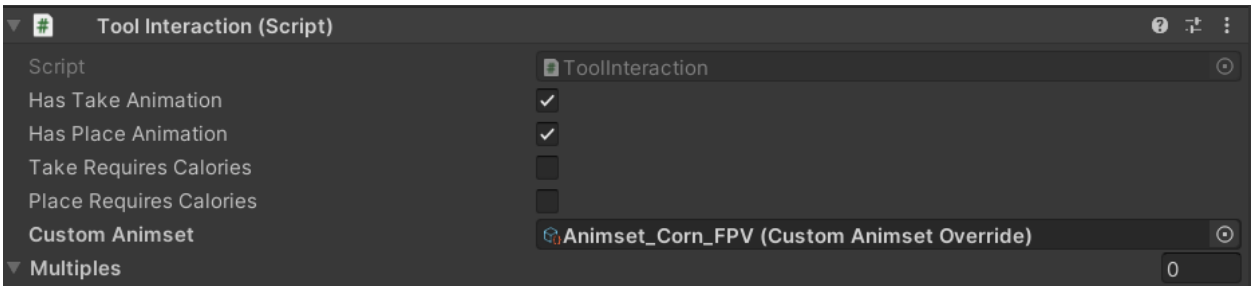
Once this is done we can turn **CornOntheCob_Eatable** into a prefab and delete it from the scene, this prefab will be useful for TPV Food, from this prefab we will create a Prefab variant renaming it in the process to **CornOntheCob_Eatable_FPV** which will, in turn, be our FPV food.



Now let's modify **CornOnTheCob_Eatable_FPV :**

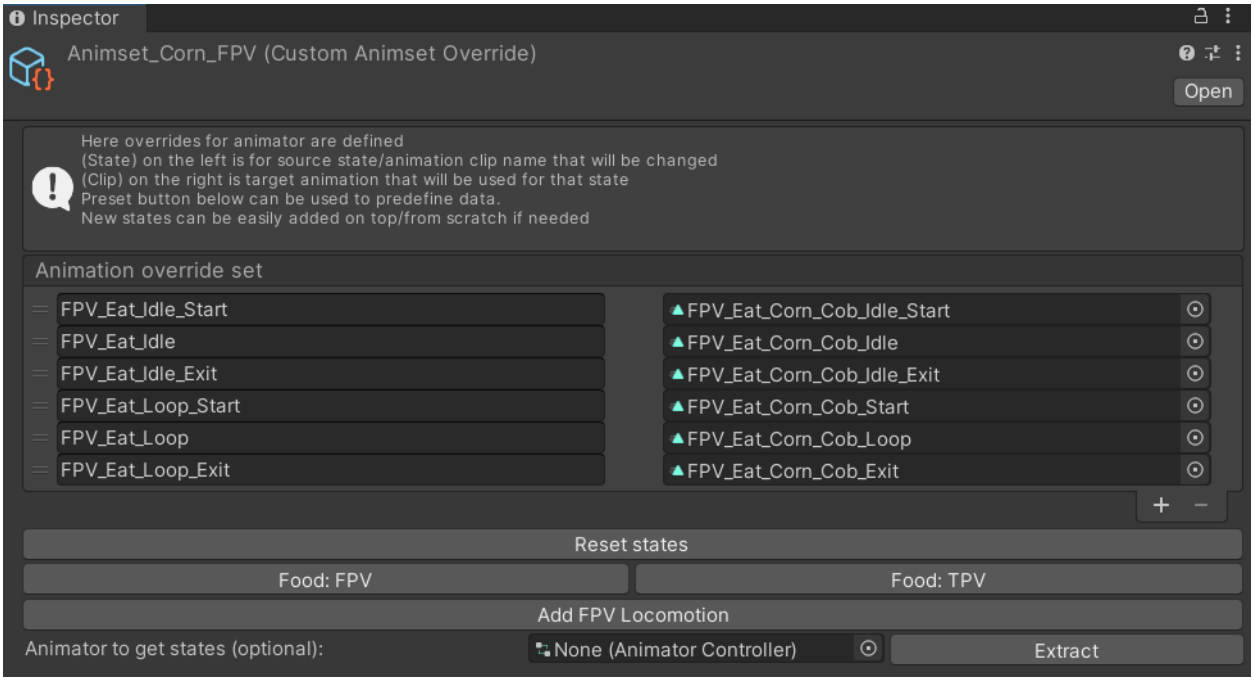
- I'll press the "Add FPV components" button which will add the **PerfomInteraction** component.
- Change **DefaultAnimationOverride** in **BiteableFoodObject** to **Custom.**
- Change **CustomAnimset** in **ToolInteraction** to a new **CustomAnimset Override** asset. Which contains the necessary FPV hand animations related to eating the Corn On The Cob.
- Lastly, I modified the position offset and rotation offset of the **BitableFoodObject** component





Overriding Custom Animations

- On the project tab right click Create → Eco → Animation → Animset Override, you will get this asset, in the example mod I choose to override the FPV animations



On the mod example, I pressed the “Food_FPV” button which will automatically add the name of the animations to override(shown in the image), and then set the correspondent animation clips.

(A reminder that these animations clips can be created by yourself via the **Avatar@FPV_Hands_tpose.fbx)**

Adding Events to Animations

After creating your **CustomAnimsetOverride** to add your own animations it is **VERY** important to add the following events to the correspondent animation clips since this is how the game actually knows the timing of when bites happen, when and what effects occur, and lastly when the food is consumed, all timed according to your animation.

Note: Without these your mod won’t work.

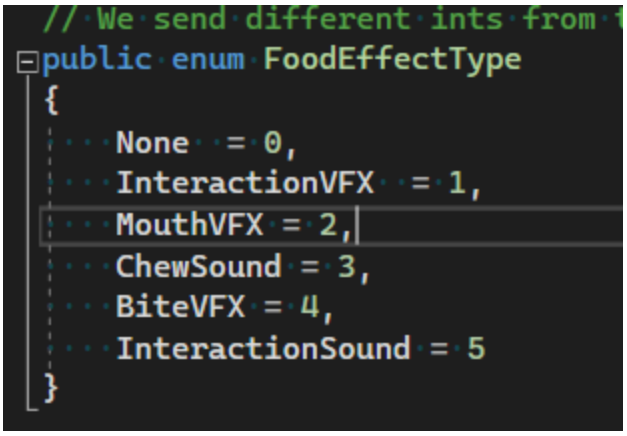
Loop Events

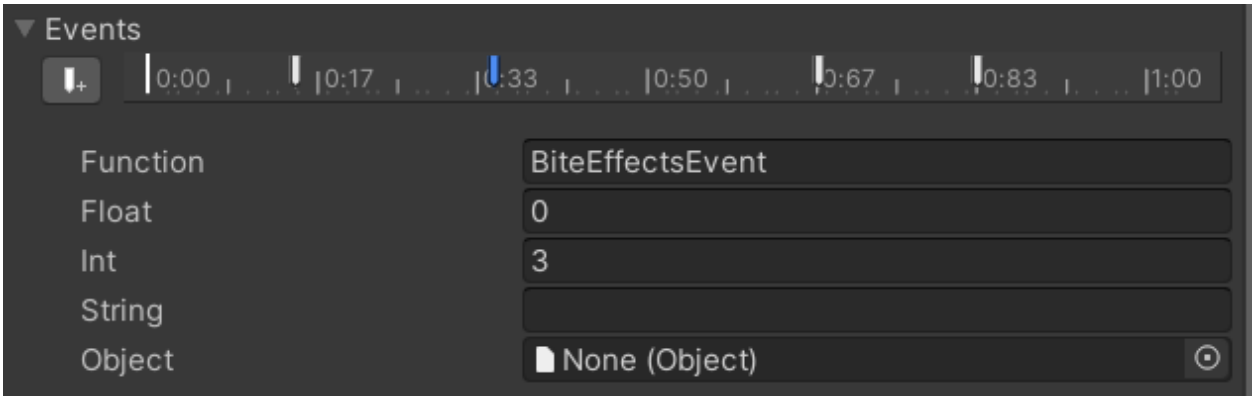
These events are primarily placed on the “**Eat_Loop**” animations whether is FPV or TPV

- **BiteEvent**: Refers to every time it takes a bite in the animations, it is done this way so you can customize and time correctly when on your loop eating

animation does the player actually takes a bite.

- **BiteEffectsEvent**: It represents multiple types of effects, it receives an integer as a parameter. The integer represents the FoodEffectType, which can be found here.

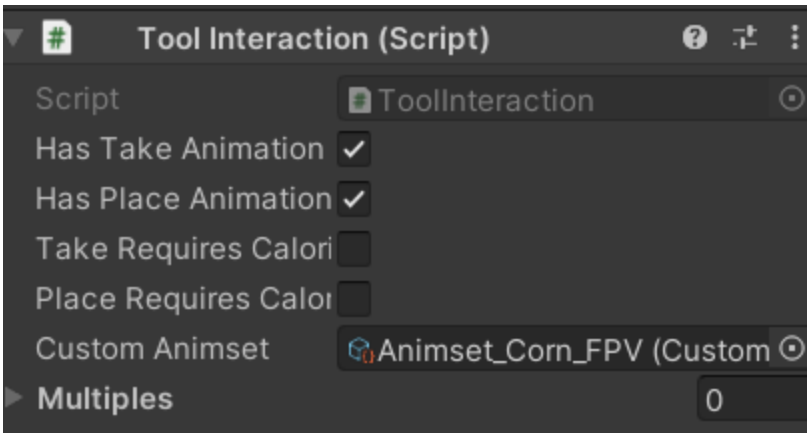




Exit Event

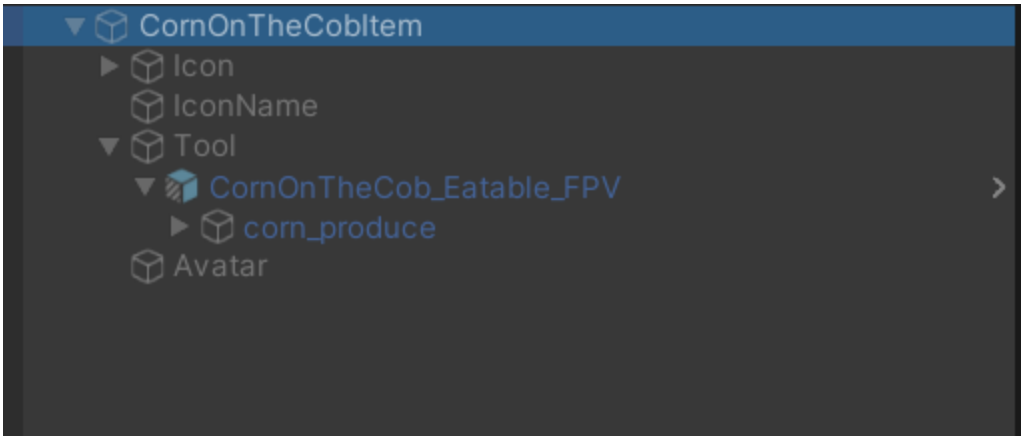
These events are primarily placed on the “**Eat_Loop_Exi**t” animations whether is FPV or TPV

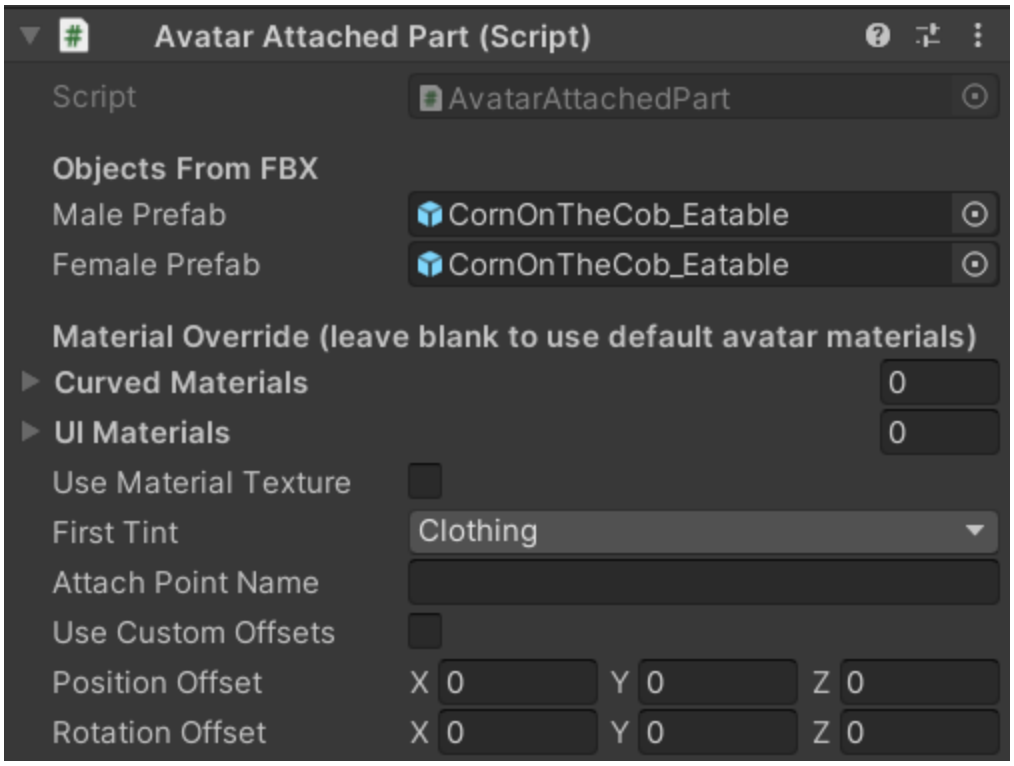
- FoodConsumedEvent: Placed after eating it determines if the BiteableFoodObject has been consumed in its entirety, adding then calories and nutrients.



Last step:

After having created the **CornOnTheCob_Eatable.prefab** and **CornOnTheCob_Eatable_FPV.prefab,** we will then place **CornOnTheCob_Eatable_FPV.prefab** as a child of Tool object , then we go to Avatar object in **AvatarAttachedPart** component and set the **Male prefab** and **Female prefab** fields both to **CornOnTheCob_Eatable.prefab.** The object under tool will be used for the FPV of the food, and the object reference in male and female fields on AvatarAttachedPart will be used for the TPV of the food.





Finally, Deactivate CornOnTheCobItem and build the modkit, then you're set to go!

Some further notes: This guide explains how the CornOnTheCob Food Mod was created and explains the critical processes involved in it that relate to food modding. It isn't therefore a strict guide on how to make your food mod. You can technically deviate from the guide, create different prefabs for the FPV and TPV, or even different prefabs for Male and Female, it's all up to the modder.

Keep in mind:

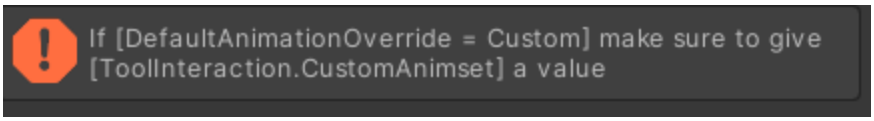
- If you want to make your custom animations make sure to add the animation events
- Name the Item correctly so that it coincides with the item class in Server
- Have the correct component on your FPV food and your TPV food.

Errors and Warnings

You might have wondered what those error signs and warning signs mean allow me to explain

- When setting the DefaultAnimationOverride to **Custom** it is expected for you to fill the field of **Custom Animset** in Tool Interaction, if this isn't the case the next

error message will appear, which indicates that you should add a **Custom Animset** because otherwise there wouldn't be an animation



- When setting the **DefaultAnimationOverride** to anything other than **Custom**, and also having the **Custom Animset** on **Tool Interaction** set, the latter one will be used instead of the specified one in **DefaultAnimationOverride,** for the warning to disappear you should change **DefaultAnimationOverride** to Custom or unset the **Custom Animset** value