Basic Documentation for Modular Anime Character Package

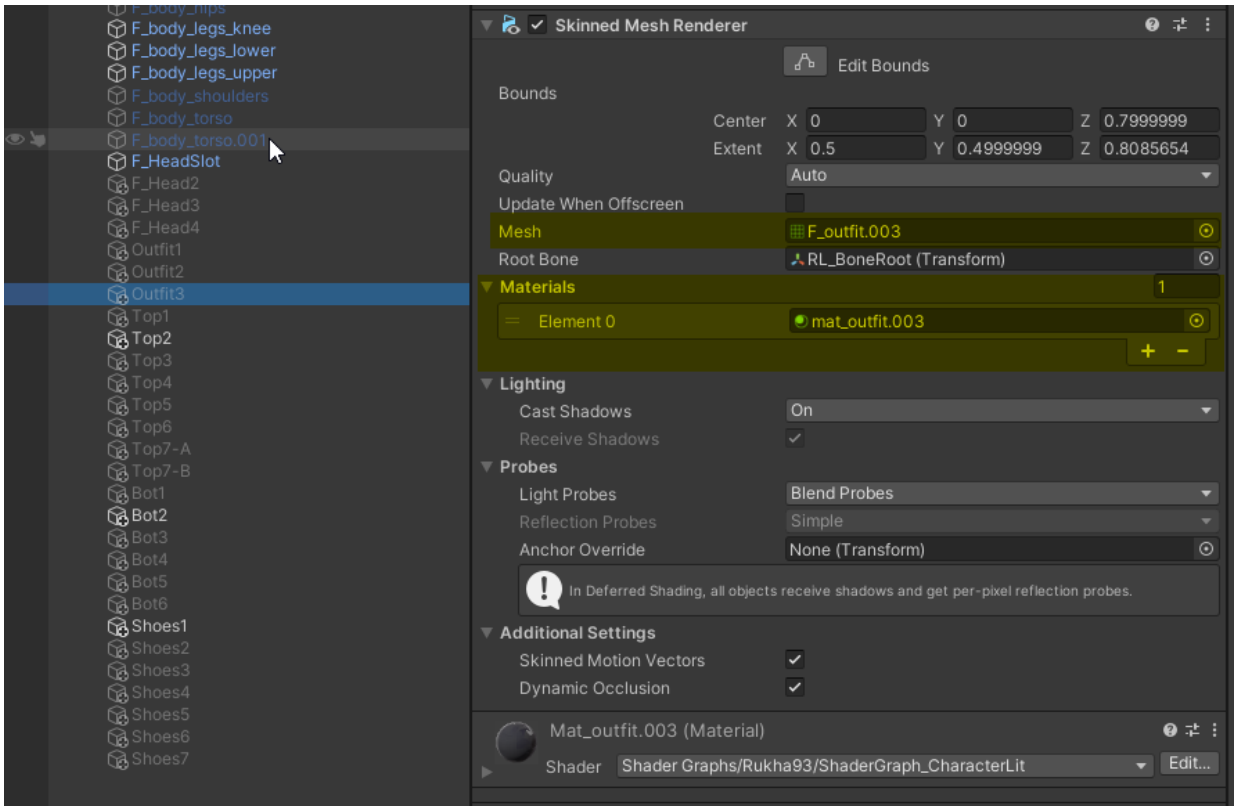
Models and customization

Method 1

The simplest way to get a custom character ready is to use the “F_FullBody AllEquipments” or “M_FullBody AllEquipments” prefabs. These prefab contains all the customizable pieces inside them, so you can just disable or remove the ones you dont want from the prefab.

Method 2

You can also set the desired mesh and material to the SkinnedMeshRenderer manually. This method is how the “F_FullBody AllEquipments” prefabs were created.
Just duplicate one of the body parts and change the Mesh and Materials in the SkinnedMeshRenderer component.
Duplicating an existing body part is important to make sure both of them are using the armature and bone hierarchy. Else the bone hierarchy will need to be initialized through code.



Method 3

The last method is the same as the second but done through script. You instantiate a new GameObject, add a SkinnedMeshRenderer component, set the new mesh and materials and copy the existing SkinnedMeshRenderer propertiers.
This method is used for the Customization Sample and can be seen in the CustomizationDemo.cs:

```
174             //instantiate new meshes, init properties, parent to character
175             GameObject go = null;
176             SkinnedMeshRenderer skinnedMesh = null;
177             foreach(var mesh in item.meshes)
178             {
179                 //instantiate new gameobject
180                 go = new GameObject(mesh.name);
181                 go.transform.SetParent(m_Character.transform, false);
182                 m_Equiped[cat].instantiatedObjects.Add(go);
183 
184                 //add the renderer
185                 skinnedMesh = go.AddComponent<SkinnedMeshRenderer>();
186                 skinnedMesh.rootBone = m_ReferenceMesh.rootBone;
187                 skinnedMesh.bones = m_ReferenceMesh.bones;
188                 skinnedMesh.bounds = m_ReferenceMesh.bounds;
189                 skinnedMesh.sharedMesh = mesh.sharedMesh;
190                 skinnedMesh.sharedMaterials = mesh.sharedMaterials;
191             }
```

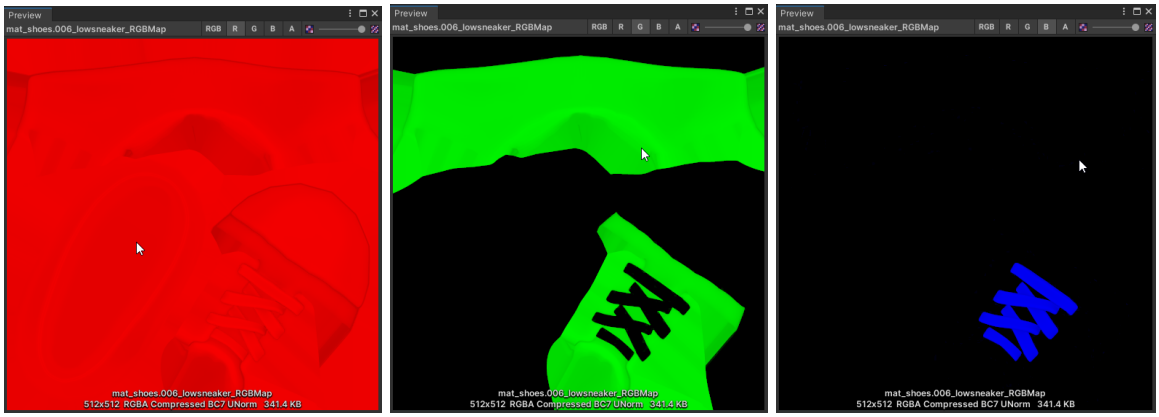
Hair

The hair prefab must be added as a child of the head bone (CC_Base_Head).

Shaders

Overview

The main shader, “ShaderGraph_CharacterLit”, uses a Mask texture where each channel defines an area of the model that can be colorized:



The ShaderGraph implementation for the color customization is in the subgraph file “SubGraph_ColorCustomization”.
It also has a customizable rim light, implemented in the subgraph “SubGraph_Rimlight”.

Variations

Besides the standard Lit version, there are 3 extra toon shaders. Though they are unlit shaders, they support one directional light to control the shadows. The implementation is found in the “SubGraph_ToonShading” subgraph.
The “SubGraph_ToonShading” subgraph uses a Custom Function to get the direction and color of the main directional light. The implementation of this custom function may need to be changed in future versions of unity. If you need help to update this, feel free to send me an email.

Basic (ShaderGraph_ToonBasic)

The Basic version is meant only for the simple areas with one single color, like the eyebrows.

Rim Light (ShaderGraph_CharacterToon)

This is version used the mask texture used for color customization.
It also adds a customizable Rim Ligh to make the model stand out.

Rim Light + Fake Outline (ShaderGraph_CharacterToon+FakeOutline)

This version simulates an outline using a dark rimlight.