Polyphoria | Unity

Unity Character Libraries

The Polyphoria Character Libraries come with a load of assets to equip a Character as you like.

Multiple arms, chests, shoes, helmets, weapons and shields are ready to be assembled to your own unique composition. Be it a medieval knight, a roman centurion, a modern age citizen, , a western gunslinger or a medieval smith – it's your choice. Each equipment uses a master material and a mask texture, which allow you to tweak its visual appearance in detail. For compatibility to other asset packs (like weapon animation sets), the delivered skeletal meshes are mapped to a Unity avatar definition.

The weapon and shield pivots are aligned to its hand grip. Make sure to adjust the Sockets to your desired needs.

The pack contains a basic functional Avatar character system which implements all of the equipment and their customization. Furthermore, we like to continue developing our shaders for the HDRP.

HDRP Skin, Eye, Hair and Tearline shaders are work in progress and will be updated accordingly.

Quick Info

Since we develop this project besides our full-time jobs and know the product by heart, it may lack some documentation every now and then. Thus, we kindly request any suggestions for assistance to enlarge the docs and especially the tutorial section for topics that need more tutorship than raw technical explanation.
Of course, if there are any questions, simply drop us a line via mail:
contact.polyphoria@gmail.com .

Documentation history

Version	Description
09.01.2022 Unity 2021.2.7f	Unity Asset store Medieval NPC

Table of content

1. Component Architecture

A complete character is a representation of multiple prefabs. These are combined on runtime into one single prefab with one skeleton.

The system is scalable and flexible to add new data.

Take a look at the scripts here:
Polyphoria/Polyphoria.Core/Scripts/Core

1.1 Modular Character

The main component is called ModularCharacter. All of our character prefabs contain a single ModularCharacter component on its root object. It contains a list of all elements this character is composed of and allows direct removal of baked parts using a custom inspector.

If your modular character is a prefab (or prefab variant) and you want to remove a baked part, you have to unpack the prefab, since Unity does not allow the removal of child elements from a prefab outside of the prefab mode. The custom inspector will check automatically, if the baked part is part of a prefab.

Property	Description
Element Parent Prefab	At runtime, each part of a modular character is baked into a single animator using a single skeleton. Each baked element is added as a SkinnedMeshRenderer using the "Element Parent Prefab" as template. For extended customization you can modify this prefab (it is located by the tag "Polyphoria_ElementParent").
Skeleton Asset	The Skeleton Asset is a base rig (provided in Polyphoria.Core) that is used to instantiate a skeleton during runtime that is used for all parts. It is intended to allow different skeletons (e.g. a male and a female variant) for different avatar types.
Modular Parts	A list of all parts this modular character contains. Displays the name of each part/prefab and a button to remove or find the prefab part from the character.

Below the baked parts, there is an empty object field titled “Drag modular prefab here to attach”. Dragging any prefab containing a ModularCharacterElement component will add this part to this modular character and store a reference in *Baked Parts*. The object field itself will remain empty to allow the subsequent addition of multiple elements. Use this to build your modular character with as many parts as you need.

1.2 Modular Character Element

A modular character can consist of an arbitrary number of elements. These elements can be anything, e.g. Clothing, Accessoires, etc.
Each element needs a Modular Character Element component on the root object. All elements provided by Polyphoria already have these added and preconfigured.
The Modular Character Element identifies a prefab as a modular part and allows it to be used in the baking process.
Additionally, it stores settings and values that influence the baking process.

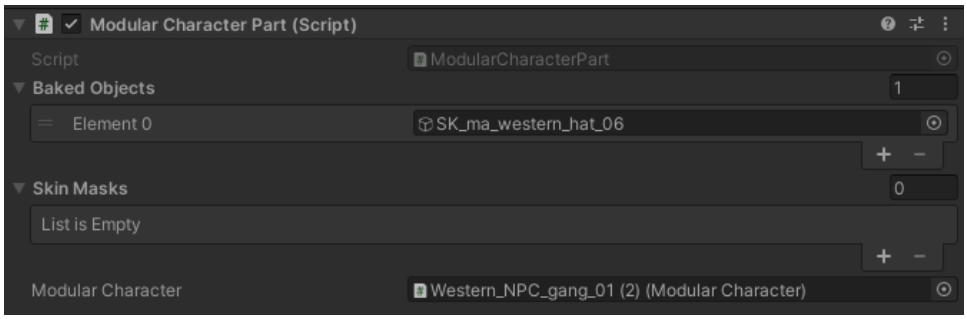
The following properties for BlendShapes are currently defined:

Property	Description
Slot / BlockedSlots	An enumeration that allows for a slot based setup. It is provided for convenience (along with supporting functions in the ModularCharacter component like "IsSlotFree" etc.) It can be used as a starting point for a custom setup. It is not required for the baking process or the base functionality.
BlendShapeTriggers / BlendShapeReceivers	Some parts contain BlendShapes that allow adjustments when combined with other parts. An example would be a hair part that has a BlendShape property "hat_equipped" to change the hairs shape when adding a hat to the character to prevent the hair from clipping through the hat model. For this functionality to work, the element that is affected (in this case the hair, that needs to change) needs to have the BlendShape name added to the "BlendShapeReceivers" list while the triggering element (in this case the hat that triggers the required change of the hair) Needs to have the same name added to "BlendShapeTriggers". Please bear in mind that currently, the editor preview of characters does not reflect the BlendShapes, since they are only applied at start and during runtime.

SkinMasks	All SkinMask textures of all baked models are merged together during runtime (and on each change of the baked parts) into a single RenderTexture and applied to any material used in the baked character that has a SkinMask property. IN short: Opacity mask to hide skinparts

Please bear in mind that currently, the editor preview of characters does not reflect the BlendShapes, since they are only applied at start and during runtime.

1.3 Modular Character Part



This component is used internally to store information when baking parts to allow a later removal of individual parts.

Property	Description
Baked Objects	After baking an element, the baked parts are stored within this list to allow a later unbaking. It is just for error checking when developing custom character elements and should not be modified directly.
SkinMasks	A reference to all used SkinMasks from ModularCharacterElements. Also needed for a later unbaking.
Modular Character	A reference to the Character these parts belong to.

1.3 Player sample Camera

We use cinemachine for this sample:
https://unity.com/unity/features/editor/art-and-design/cinemachine

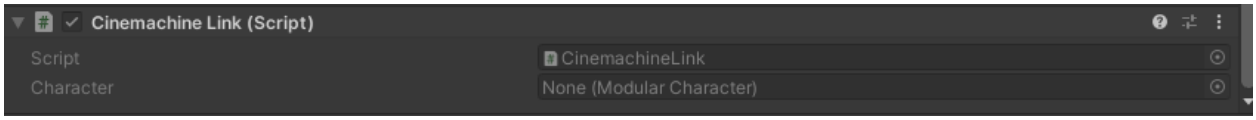
Game cameras are a delicate topic and complex requirements on gameplay or project functionality requires a full implementation on customers side.
But we do believe a exemplary camera can help as a starting point to showcase our character costumes.

#1 Drag the Virtual Camera prefab in the szene, found in

Assets\Polyphoria\Polyphoria.Core\Scripts\Helper\ThirdPersonVirtualCamera

Into the scene.

#2 Select your linked character into the Cinemachine Link script

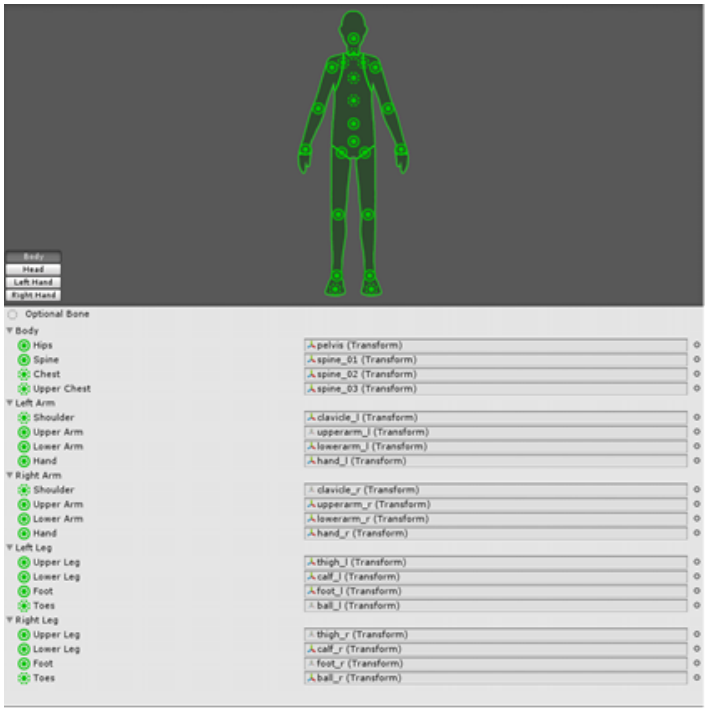


#3 Hit the play button and the camera will align to the selected character

2. Assets

Since the core of our Character libs is visual and easy to explore, there is not much more to say, than that they are located in their respective folders: Meshes, Textures and Materials. The subfolders are self-descriptive, see for yourself. Well, there is some information… ;-)

2.1 Meshes



Arms, chests, shoes, helmets and (to make it complete) a single head come **as skinned meshes** and are mapped to a standard Unity humanoid avatar definition. Therefore, they should work with any animation targeted for the unity humanoid avatar.

All meshes are created as parts of the same character, so they perfectly fit together with each other component.
On the other hand, Attachments like weapons and shields come **as meshes**. Their pivot is aligned to the hand grip of the skeleton and two placed sockets (see SK_RomanDefaultCharacter → root → … → LeftHandSocket / RightHandSocket).

2.2 Textures

Each mesh/Material has a set of textures consisting of:

Texture Type	Suffix
Albedo / Diffuse	_D
Normal	_N
Metallic (R), Smoothness (G), Ambient Occlusion (B)	_MSO
Tint Mask (RGBA channels explained in the Master Material section)	_ID

2.3 Shader/Materials

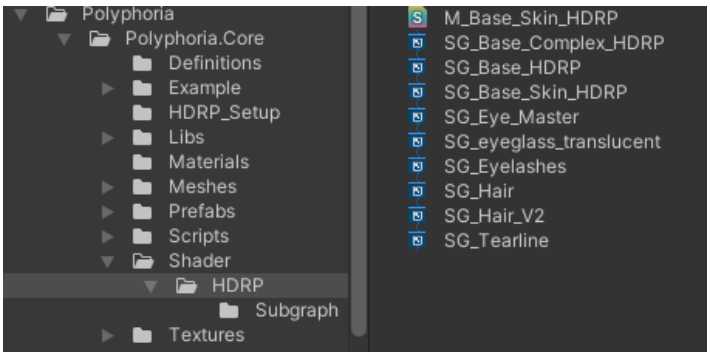
Each mesh has a default material assigned and various Tint Sample material instances. More details in the next chapter.

Master Material

The idea behind the material structure is as follows, may slightly change if needed:

All Materials use the M_Base.shader that has been designed using the Shadergraph/Amplify Shader Editor. All shader function sources have been included, if you own the Amplify Shader Editor, you can open and edit it conveniently.

- M_Base explained

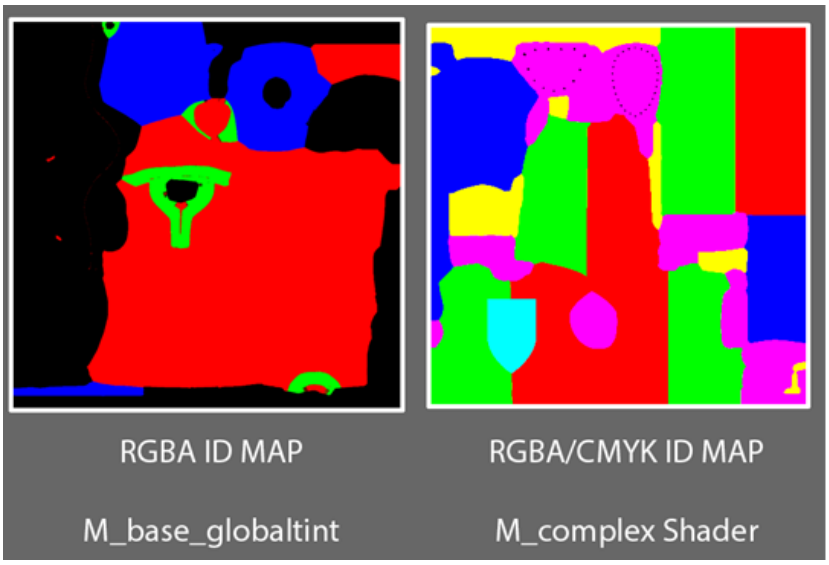


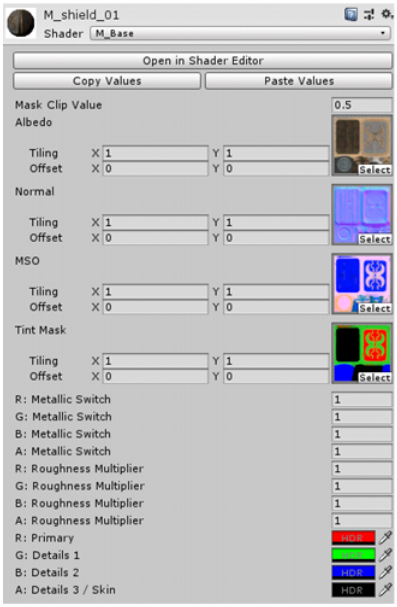
Overall this material is no rocket science. At first you have the Albedo, Normal and MSO textures exposed to material instances to set them for each equipment individually.
The actual magic happens inside of the highlighted material functions. Since the material graph explains everything, here is more of an overview:

- RGBA_Tint
  This function exposes the *Tint Mask* texture and *Tint Mask - Color* parameter group, which tints the material. Each parameter:
  - *R: Primary*
  - *G: Details 1*
  - *B: Details 2*

- *A: Details 3*

Tints the respective channel of the *Tint Mask* texture. With the alpha value of each tint color you can control their tint intensity. Hint: Use a color value greater than V=1.0 if you want to have brighter color tints.





- RGBA_Switch

This function exposes the *Tint Mask* texture and *Tint Mask - Metallic* parameter group, which enables or disabled the usage of the metallic texture, respectively to the *Tint Mask* texture channel.

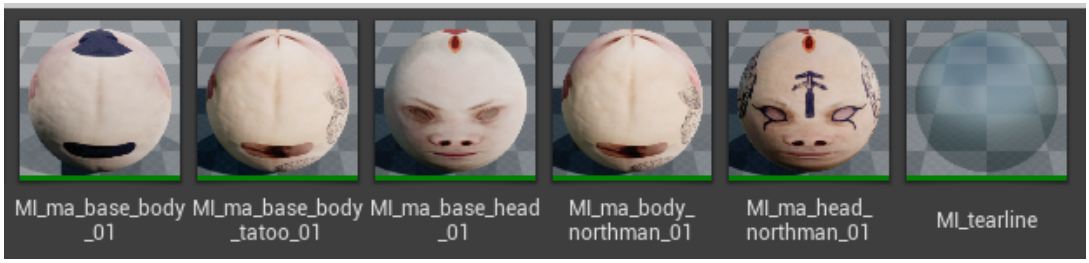
- RGBA_Multiplier

This function exposes the *Tint Mask* texture and *Tint Mask - Roughness* parameter group, which controls the roughness of the respective to the *Tint Mask* texture channel.

SkinColor

Can be changed by assigning the delivered Material instances. The provided Material samples vary between projects.

We offer as a starting base:
SG_Base_Skin_HDRP



3.0 FAQ

Thank you for your trust and support!

3.1 **Render pipelines:**

In case you find some problems with our HDRP setup feel free to let us know to streamline our development better for Unity as well.