Character Wound System

Documentation



V1.2.0

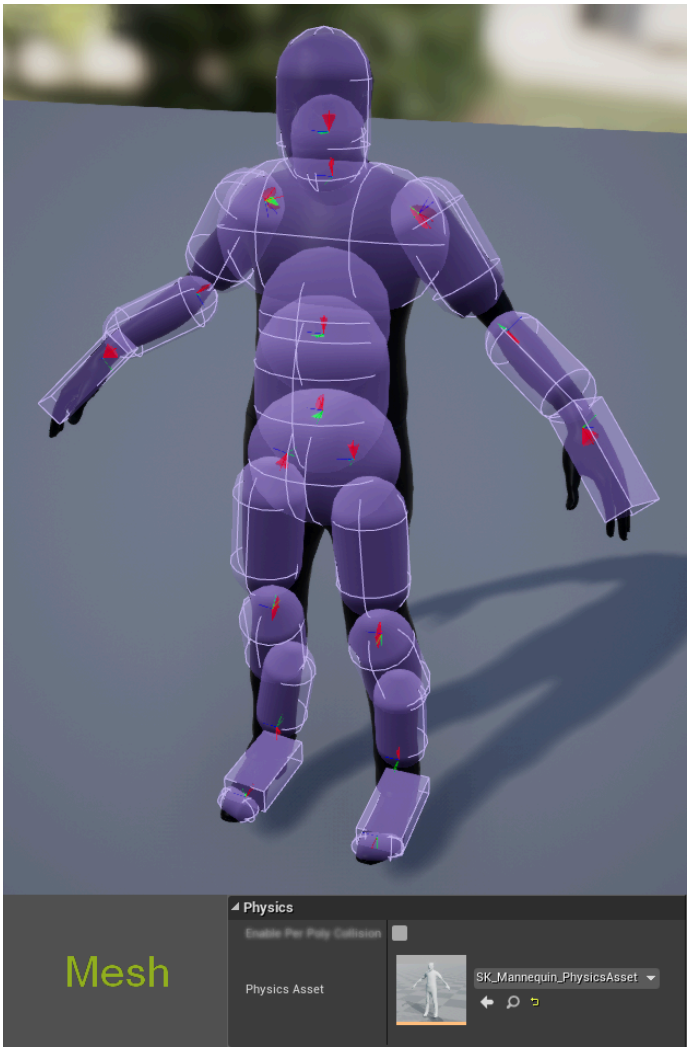
Table of contents:

- Creating a new compatible character
  - Character Physics Asset
  - Methods
  - Examples:
  - BloodMask (WoundComponent) implementation
    - Character mesh UVs
    - Character Materials
    - Character Blueprint
  - ProjectedWound (ProjectedWoundComponent)
    - Character Materials
    - Character Blueprint
  - BloodSoakingSpot (BloodSoakingSpotComponent)
    - Character Materials
    - Character Blueprint
  - Combined methods
- Integration
  - Player/Weapon usage
- Extras
  - F.A.Q.:
    - BloodMask
  - Changelog:

Creating a new compatible character

Character Physics Asset

You need to have a Physics Asset assigned to your character mesh, that has most of its primitive components as close to the mesh as possible, as the physics asset is used for line tracing a hit, and if it hits too far from the mesh the wound will be smaller or not visible.



Character PhysicsAsset and having it assigned in your mesh asset

Methods

This pack includes 3 different methods for wounds:

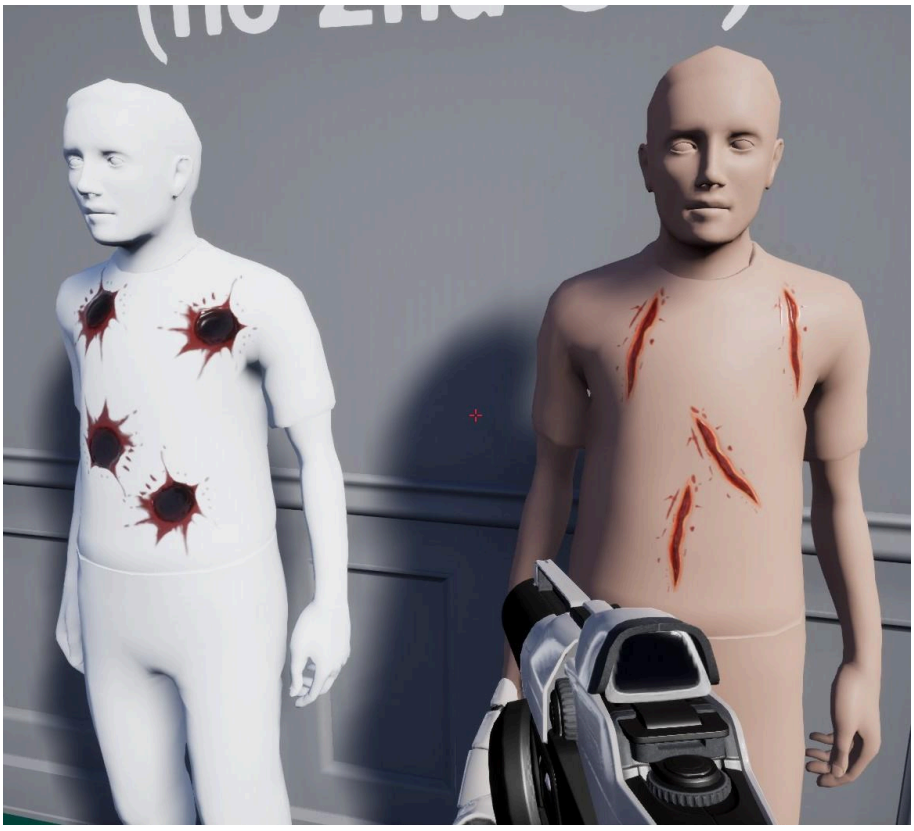
- **BloodMask** (WoundComponent): It creates a blood texture mask that can be “painted” on in two different ways, by using a sphere or by projecting a wound mask into it. It have very little runtime performance cost as this is just an extra texture in your materials, but it renders to texture when a new wound is made and requires your character to have a second UV



BloodMask / BloodMask using bullet holes textures projected in the blood mask

-

- **ProjectedWounds** (ProjectedWoundComponent): Similarly to decals (but it's not a decal), it projects a wound texture into the character mesh it's more detailed and have a normal map, it does have a performance cost as it's calculated in the material shader and have a limit of how many wounds you can make (up to 5 by default) but does not require a second UV



- **BloodSoakingSpot** (BloodSoakingSpotComponent): this is an effect that makes character clothing soak with blood over time, it's similar to the ProjectedWound as it's calculated in the material shader, but is cheaper as this is just a sphere combined with some masks (up to 5 by default)



- \+ you can combine the SoakingWound to the other two methods

Examples:

There's some example files for the characters that you can use as a reference for implementing it in your characters.

These are found in "/CharacterWoundSystem/Demo/Blueprint/Characters/":

- BloodMask: **CH_BloodMaskExample** / **CH_BloodMaskProjectedExample**
- ProjectedWounds: **CH_ProjectedWoundExample** / **CH_ProjectedWoundKnifeExample**
- BloodSoakingSpot:
    - Standart: **CH_BloodSoakingExample**
    - Combined with BloodMask: **CH_BloodMaskSoakingExample**
    - Combined with ProjectedWounds: **CH_ProjectedWoundSoakingExample**

BloodMask (WoundComponent) implementation

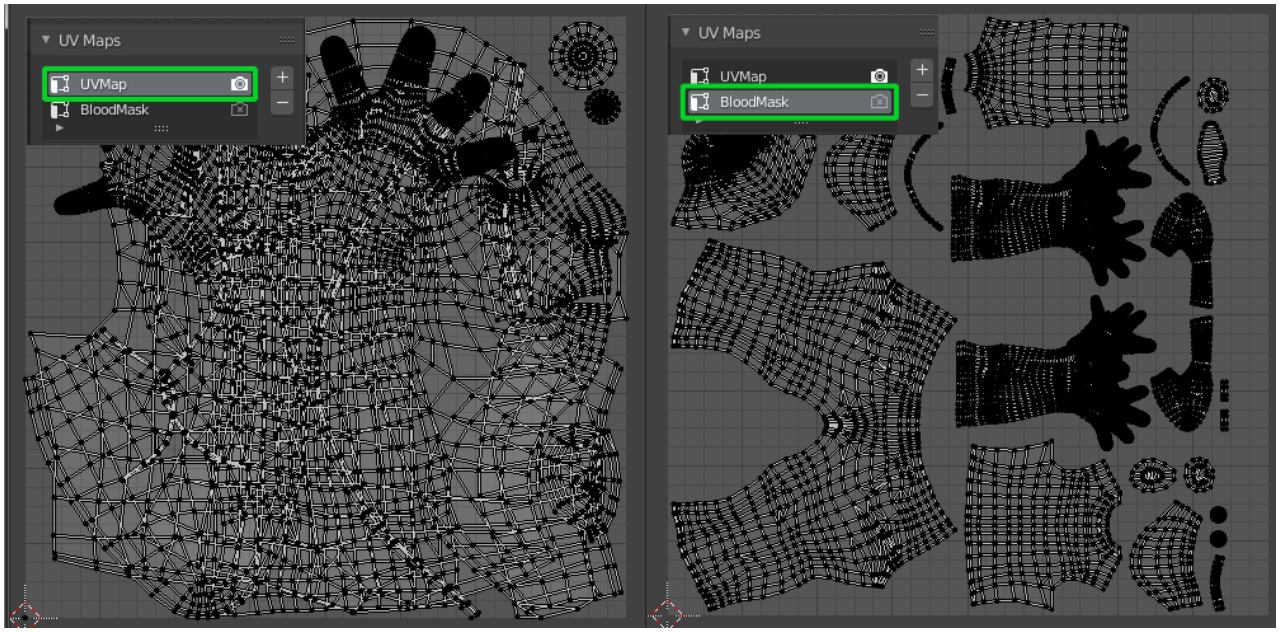
Character mesh UVs

This method requires you to have a second UV in your character mesh, just like lightmap UVs used for baking lighting in StaticMeshes.

In this case, UE4 doesn't offer any automatic lightmap UV creation for SkeletalMeshes, so you need to create this in your DCC software.

In the screenshot below, it shows that the first UV channel uses the normal unwrapping configuration as you would normally do, and the second channel is an unwrap that tries to:

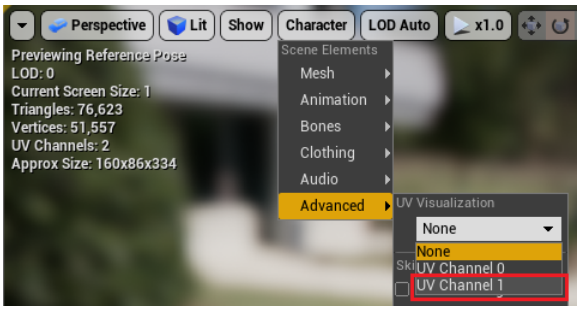
- minimize seams, to keep it as contiguous as possible
- cover most of the UV space but also have an extra space for padding between UV islands
- the full body, even if using multiple materials, share the same UV space, with no overlapping UVs. (you also should not be using the mirror modifier as you need to have a unique UV coordinate for each section of your mesh)



Screenshot using Blender DCC software

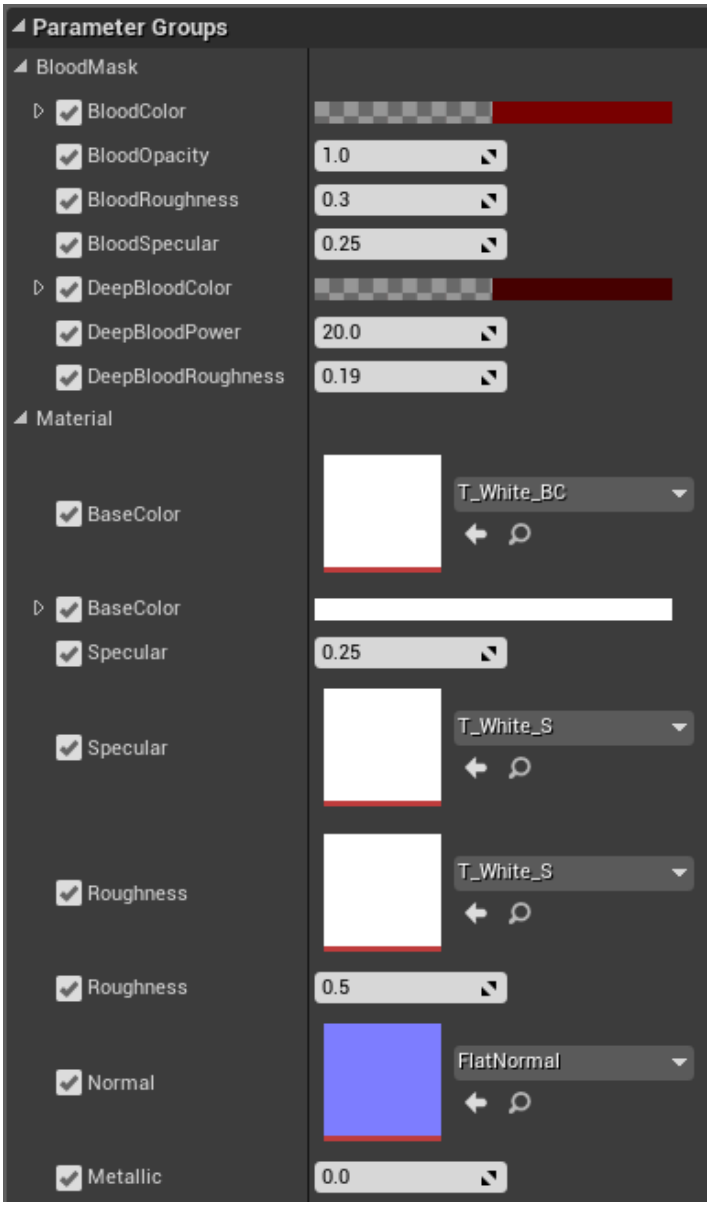
Note: for more information on this subject, visit: *https://docs.unrealengine.com/en-US/Engine/Content/Types/StaticMeshes/LightmapUnwrapping/index.html#customlightmapuvs*

To make sure that your mesh have the second UV in UE4: (UV Channel 1)



Character Materials

Your character needs to use materials that somehow use the generated blood mask texture, so in the included files you can use "**M_BloodMask**" as the Parent in your Material Instances, as this material already implements it so it's easier to use.

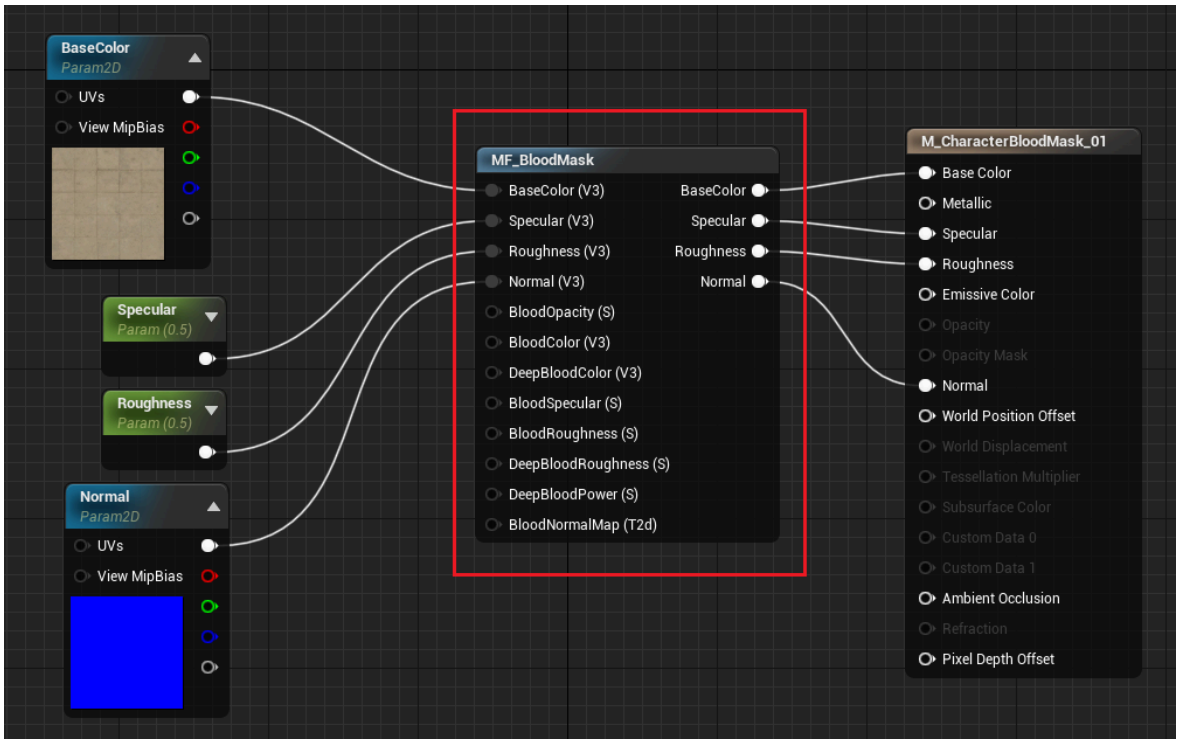


By using **"M_BloodMask"** you can already customize plenty of properties:

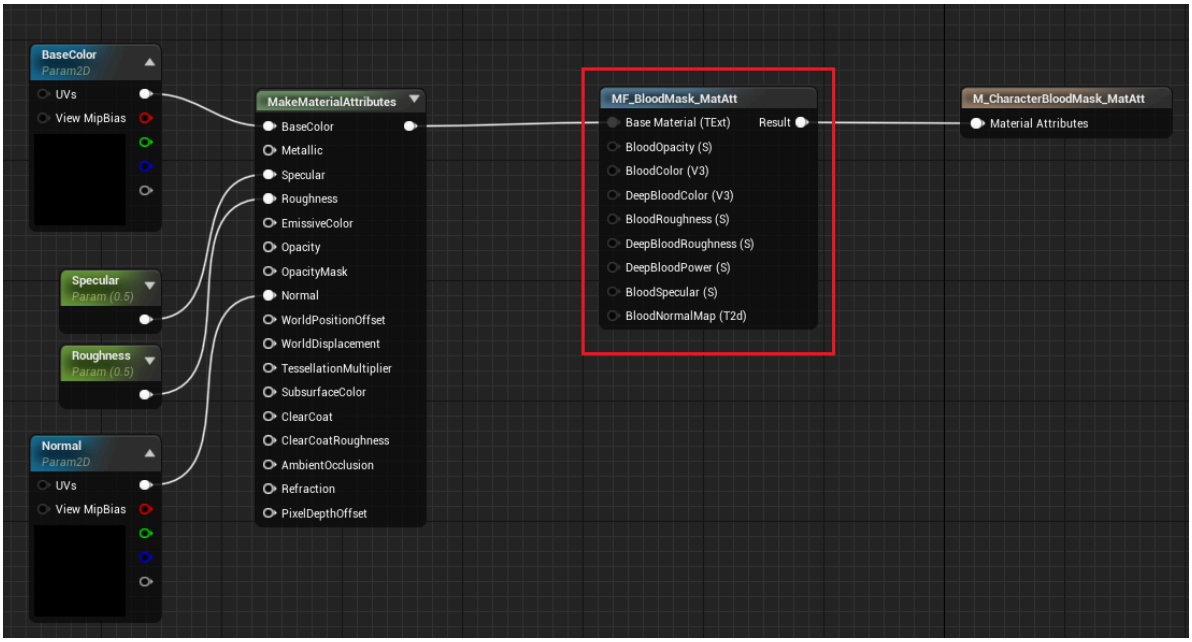
- **BloodColor**: the blood color
- **BloodOpacity**: how much blood is visible in this material
- **BloodRoughness**: roughness just in the blood parts
- **BloodSpecular**: specular just in the blood parts
- **DeepBloodColor**: the blood color in the high opacity areas
- **DeepBloodRoughness**: roughness just where the blood mask have a high opacity (multiple hits)
- The rest of the properties are just generic material options that you can put your character textures

And if you want to add the blood mask to your character materials, there's two MaterialFunctions you can use, depending on the material type you are using:

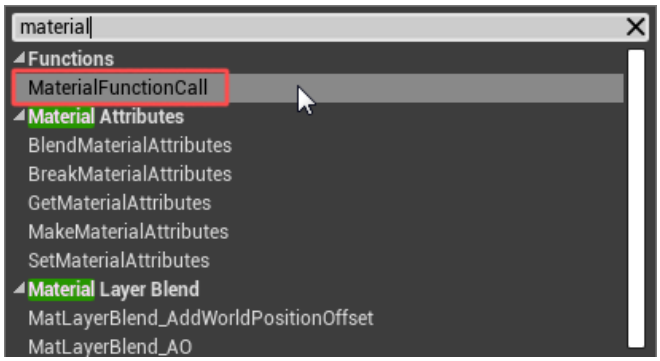
- **MF_BloodMask**: this node modifies your BaseColor, Specular, Roughness and Normal, and you can add new parameters to the properties you want to change:



- **MF_BloodMask_MatAtt**: similar to MF_BloodMask, this one uses MaterialAttributes mode:



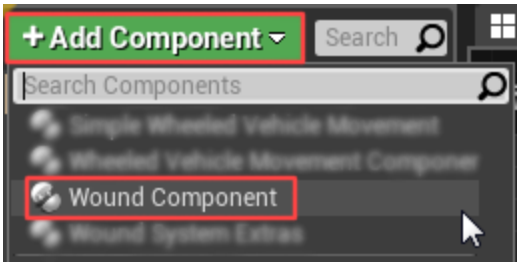
To add these nodes you first add a MaterialFunctionCall and then select the function name in the details panel:



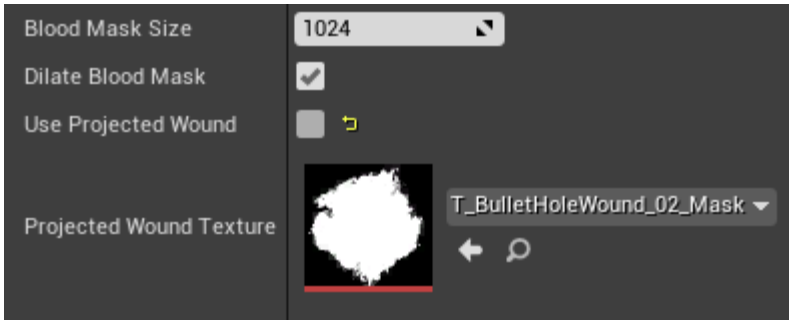
Character Blueprint

These are the steps you need to follow to make your character mesh work in this method:

1. Create a Character blueprint, or use any character blueprint you already have
2. Open your character blueprint, and add a new "**WoundComponent**" component:

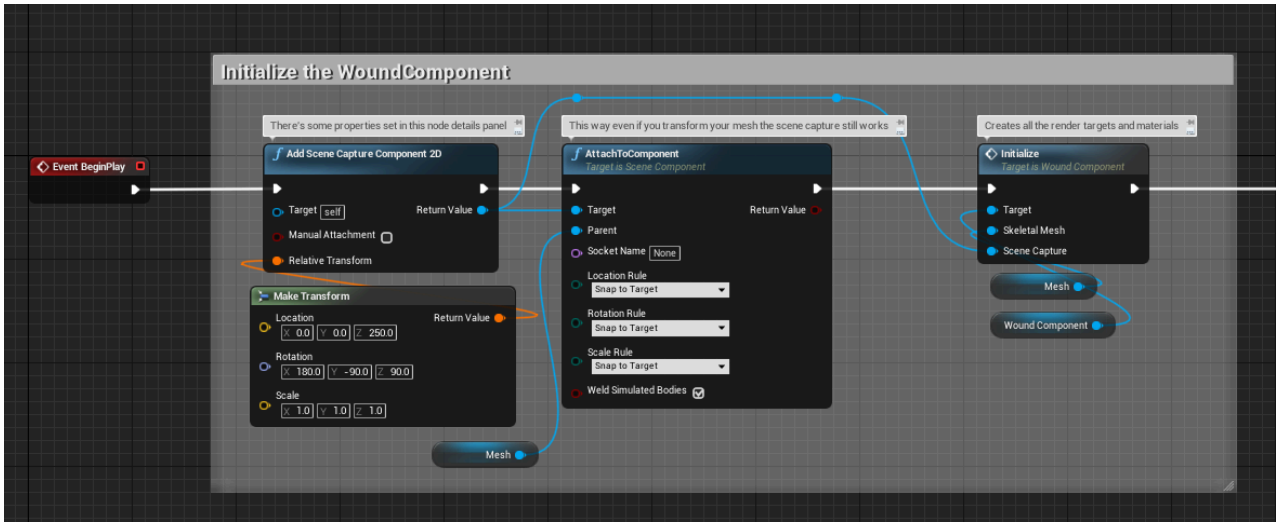


3. You can select the component in the list and in the details panel you can choose a custom resolution for the texture mask (power of 2):

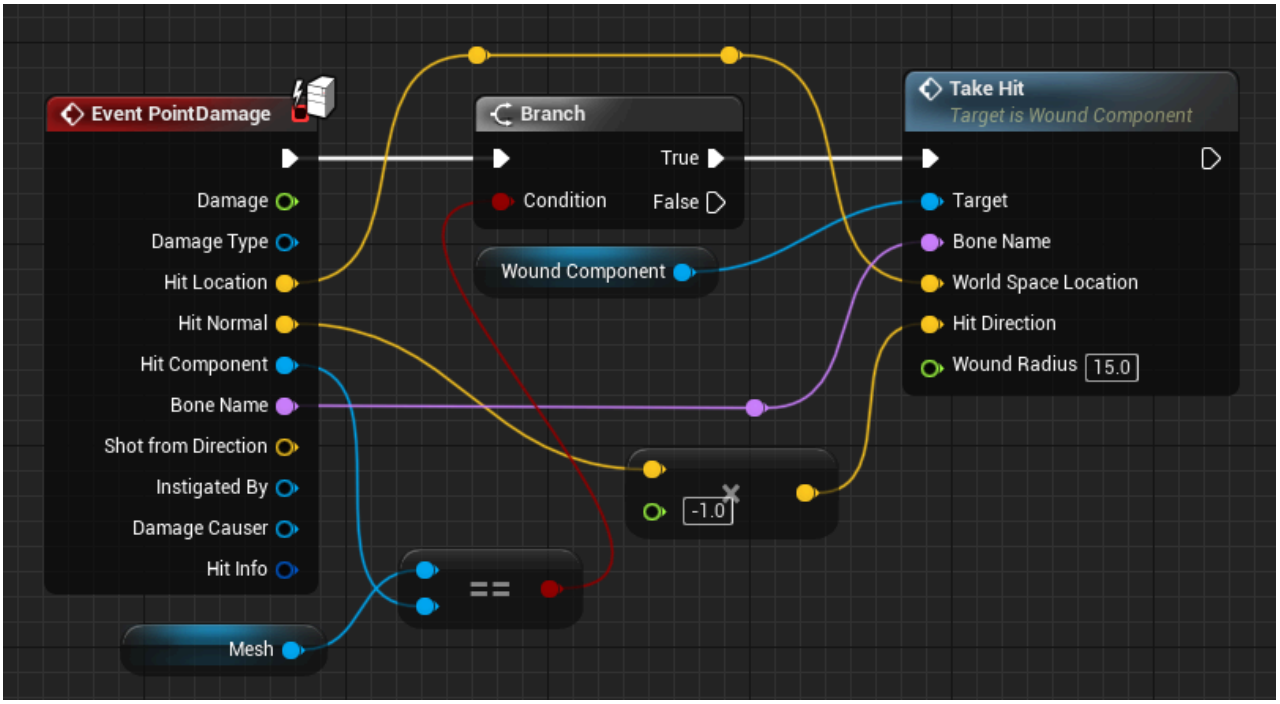


4.

5. You need to copy some code from **CH_BloodMaskExample**, find the BeginPlay event and copy this to your character BeginPlay:



6. Add a PointDamage to your character and call the TakeHit method from the WoundComponent:

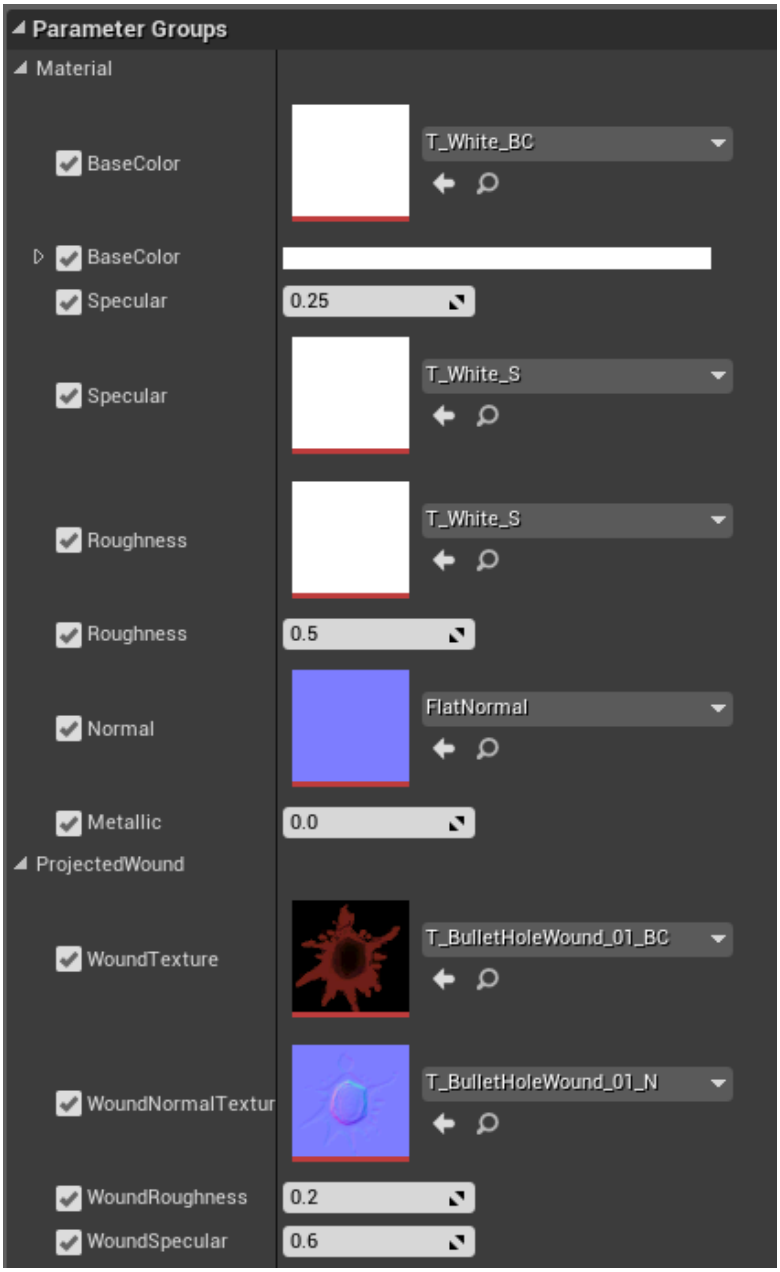


7. Select the “Mesh” component and in details panel select your mesh

ProjectedWound (ProjectedWoundComponent)

Character Materials

Your character needs to use materials that include the material nodes that use the projected wounds shader code, you can use "**M_ProjectedWounds**" as the Parent in your Material Instances, as this material already implements it so it's easier to use.

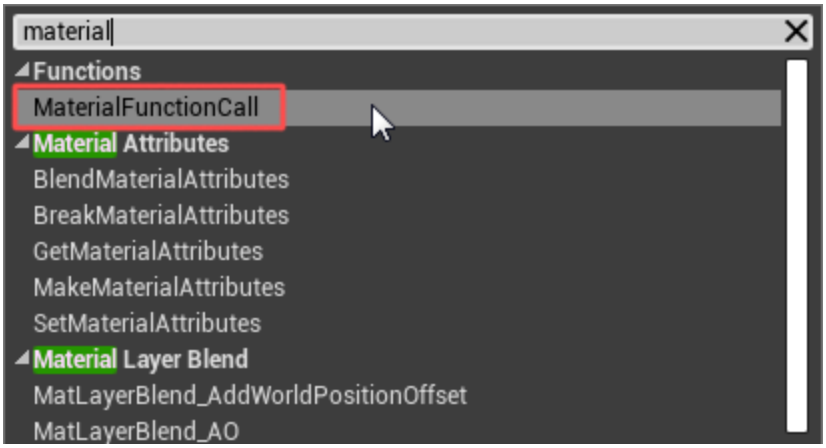


By using **"M_ProjectedWounds"** you can customize some properties:

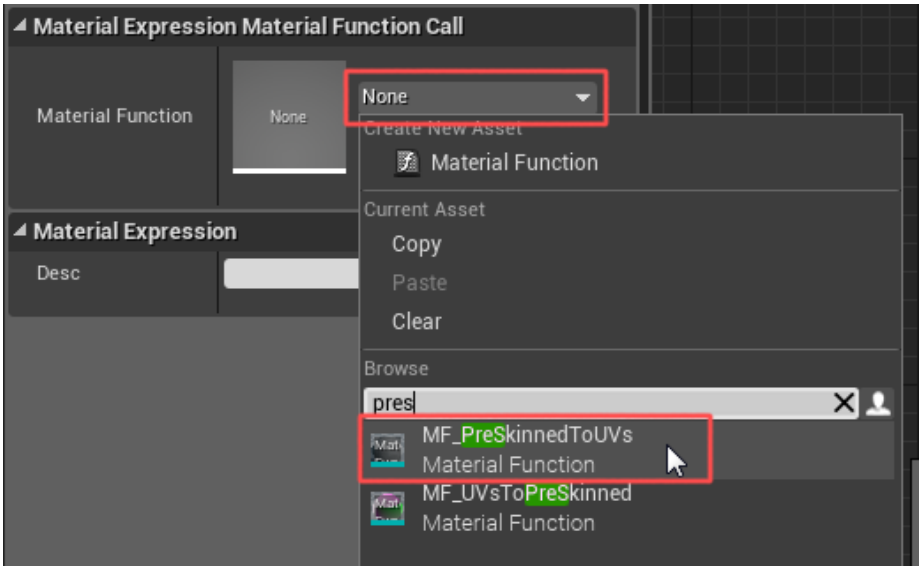
- **WoundTexture:** The texture used as the wound texture (overwritten by the component)
- **WoundNormalTexture:** The texture used as the wound normal map texture (overwritten by the component)
- The rest of the properties are just generic material options that you can put your character textures

And if you want to add the projected wounds to your character materials, these are the steps you need to follow:

1. Without having any node selected, in the details panel expand the material section and then set "**Num Customized Uvs**" to **6**, this will add some new parameters at the bottom of your material main node
2. Add two MaterialFunctions:

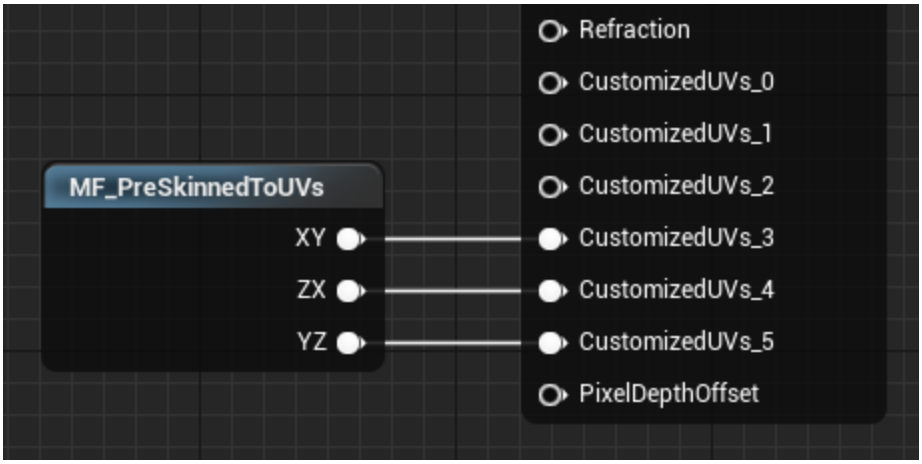


3. In one of them, in the details panel search and select "**MF_PreSkinnedToUVs**":



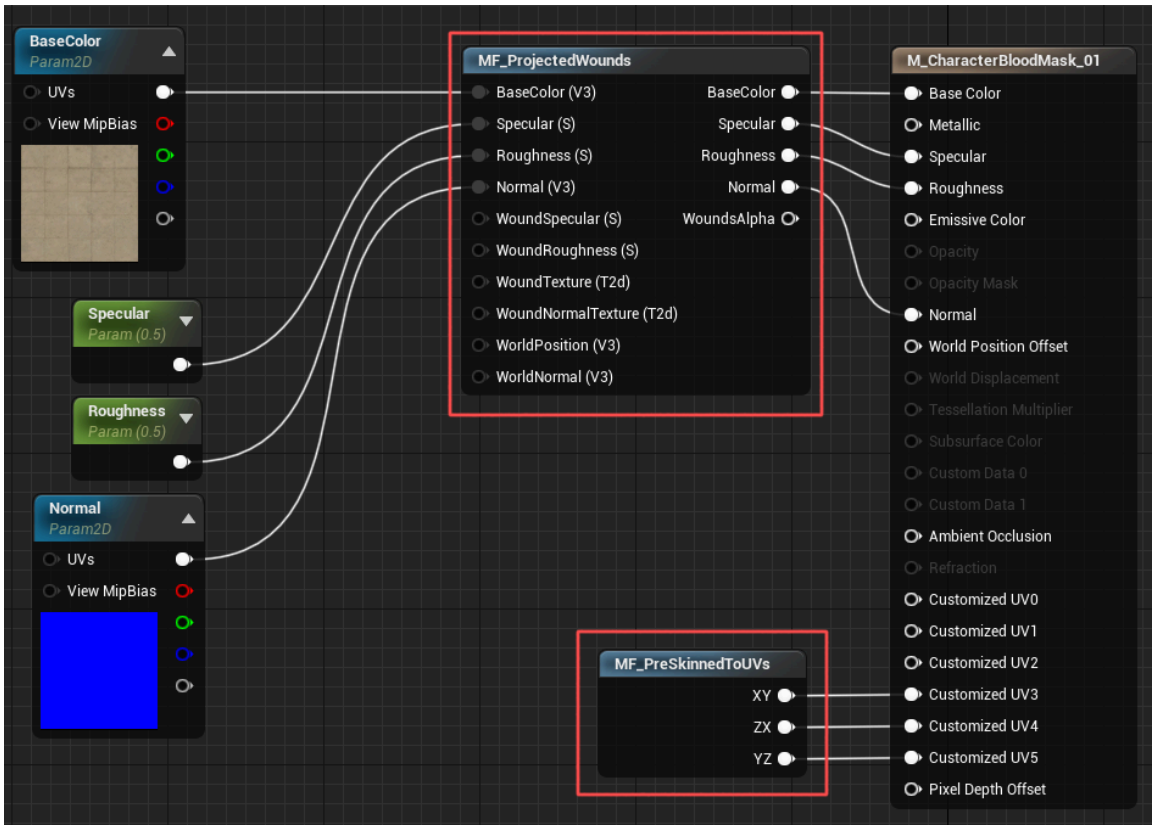
4.

5. Then connect this node to the main material node like this image:



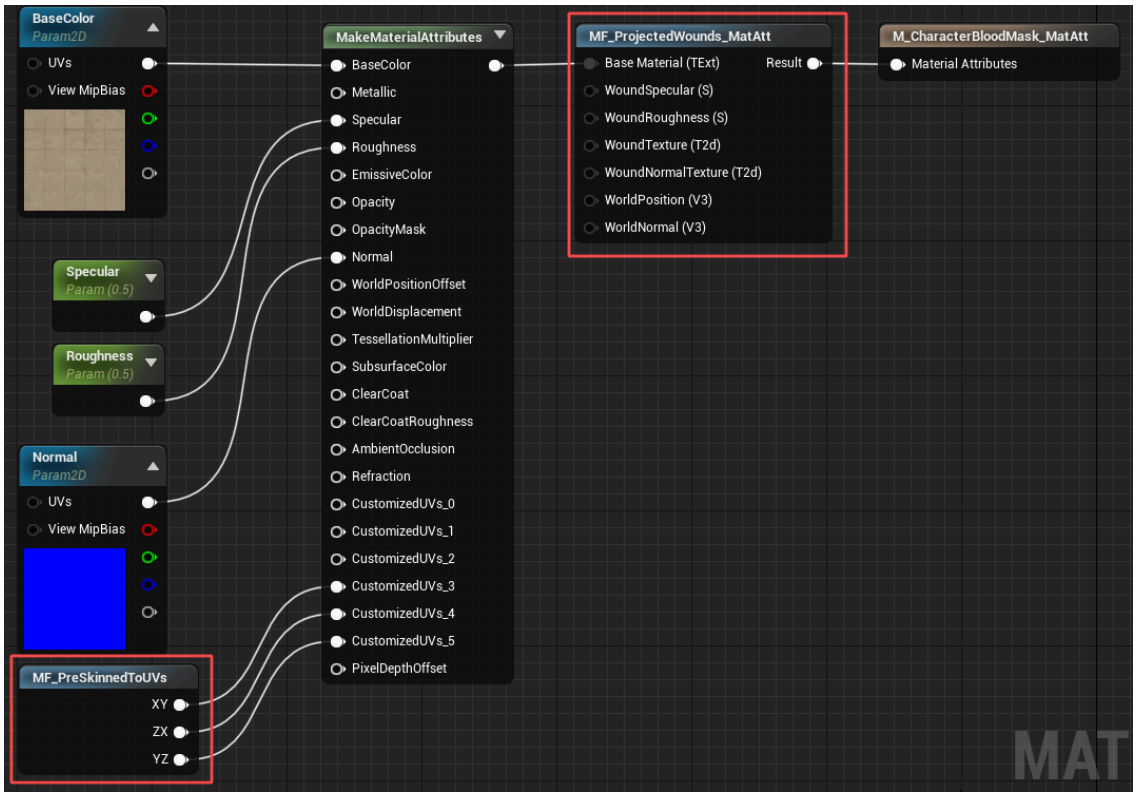
6. And finally there's two MaterialFunctions you can use, depending on the material type you are using:

- **MF_ProjectedWounds**: this node modifies your BaseColor, Specular, Roughness and Normal, by adding the projected wounds on top of your materials, and you can add new parameters to the properties you want to change:



-

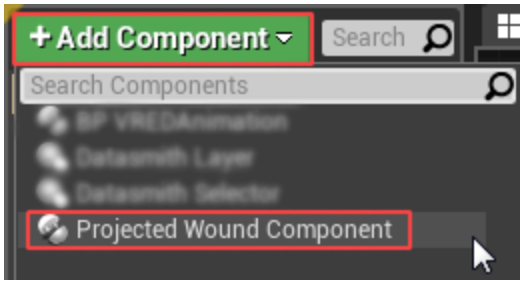
- **MF_ProjectedWounds_MatAtt**: similar to MF_ProjectedWounds, this one uses MaterialAttributes mode:



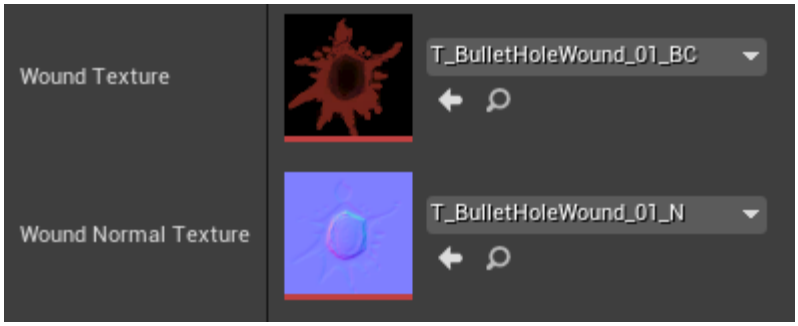
Character Blueprint

These are the steps you need to follow to make your character mesh work in this method:

1. Create a Character blueprint, or use any character blueprint you already have
2. Open your character blueprint, and add a new "**ProjectedWoundComponent**" component:

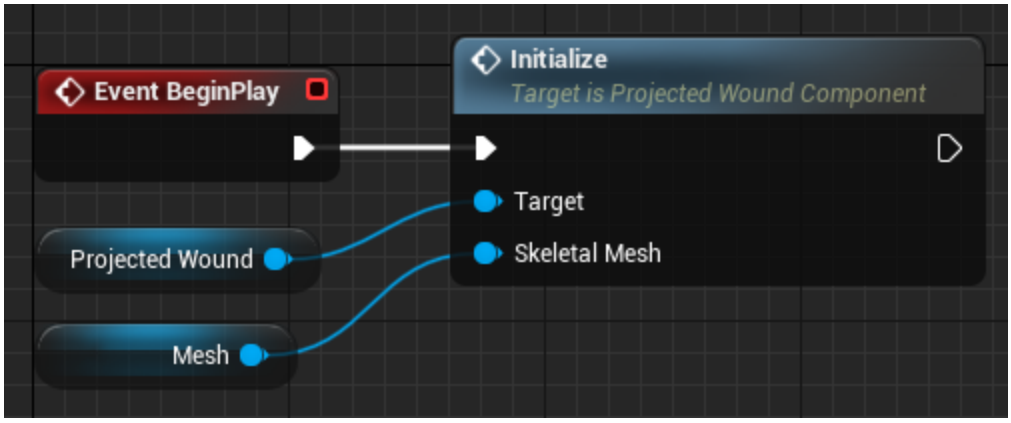


3. You can select the component in the list and in the details panel you can choose the WoundTexture and WoundNormalMap that is going to be used in your character: (you need to have the texture parameters with the same names in your materials for this to work, otherwise you can have your own textures for each material for example a wound texture for skin/cloth)

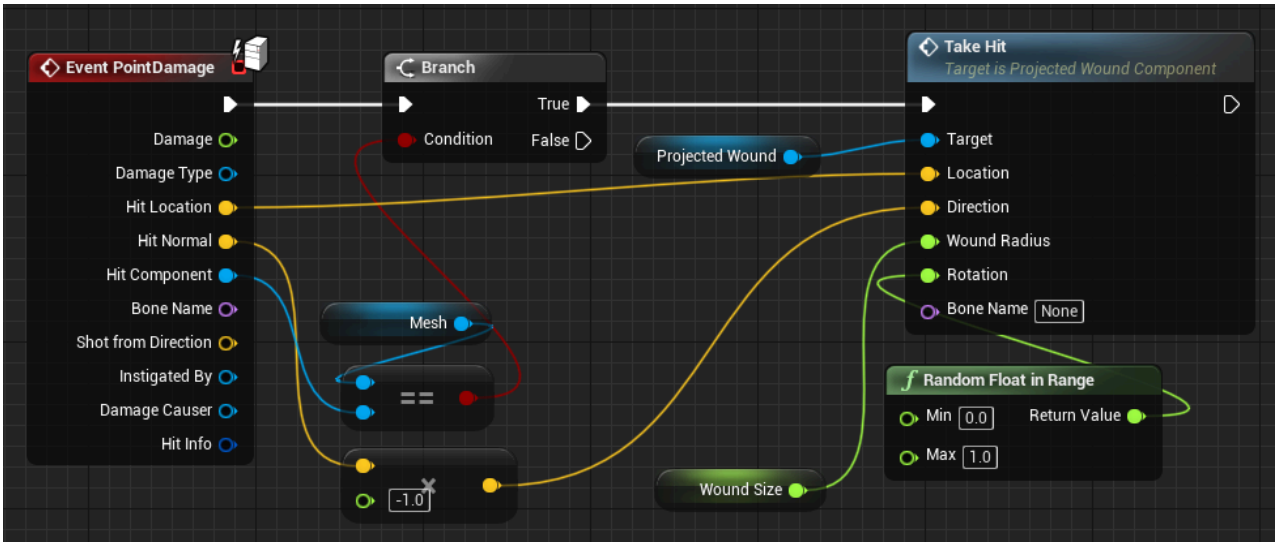


4.

5. Find the BeginPlay event and add this to your character BeginPlay:



6. Add a PointDamage to your character and call the TakeHit method from the ProjectedWoundComponent:

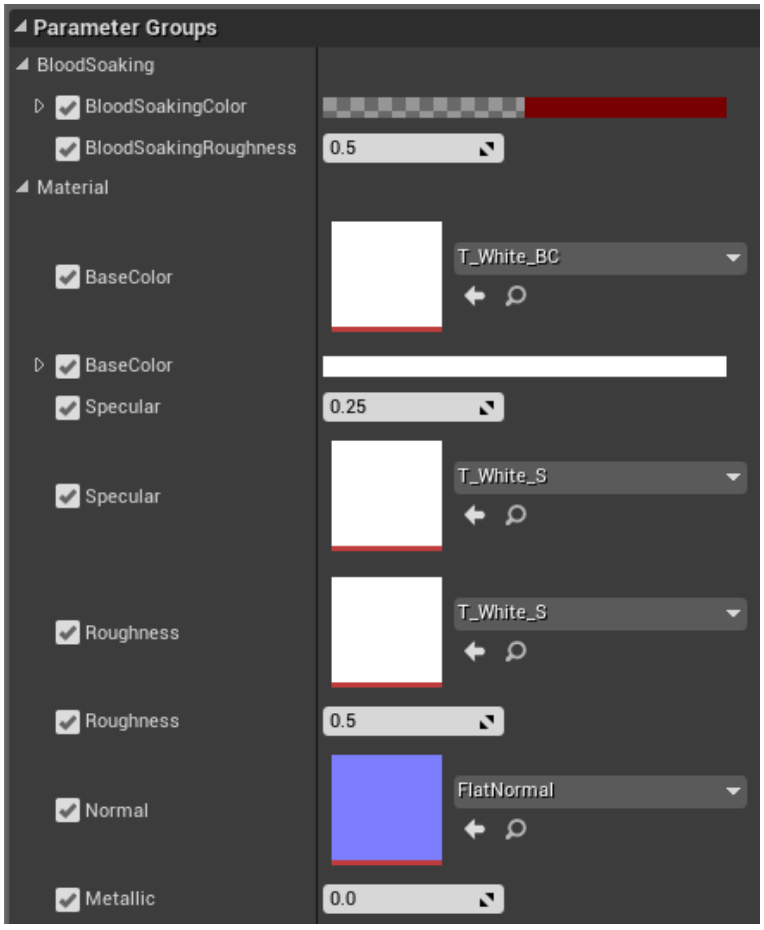


7. Select the “Mesh” component and in details panel select your mesh

BloodSoakingSpot (BloodSoakingSpotComponent)

Character Materials

Your character needs to use materials that include the material nodes that use the blood soaking shader code, you can use “**M_BloodSoaking**” as the Parent in your Material Instances, as this material already implements it so it’s easier to use.

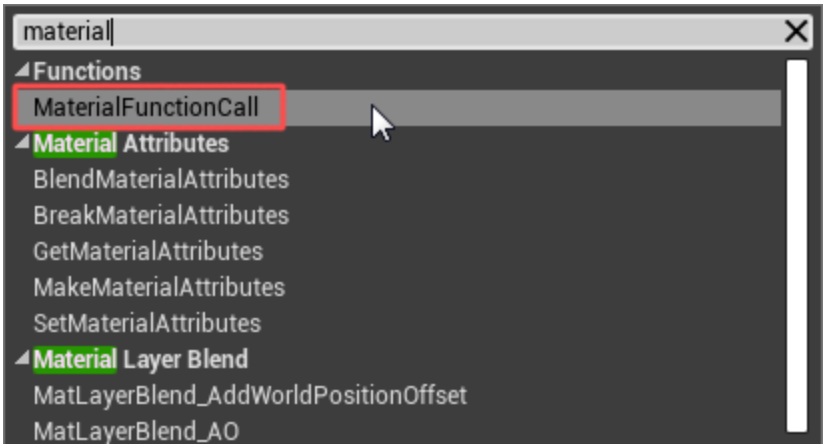


By using **“M_BloodSoaking”** you can customize some properties:

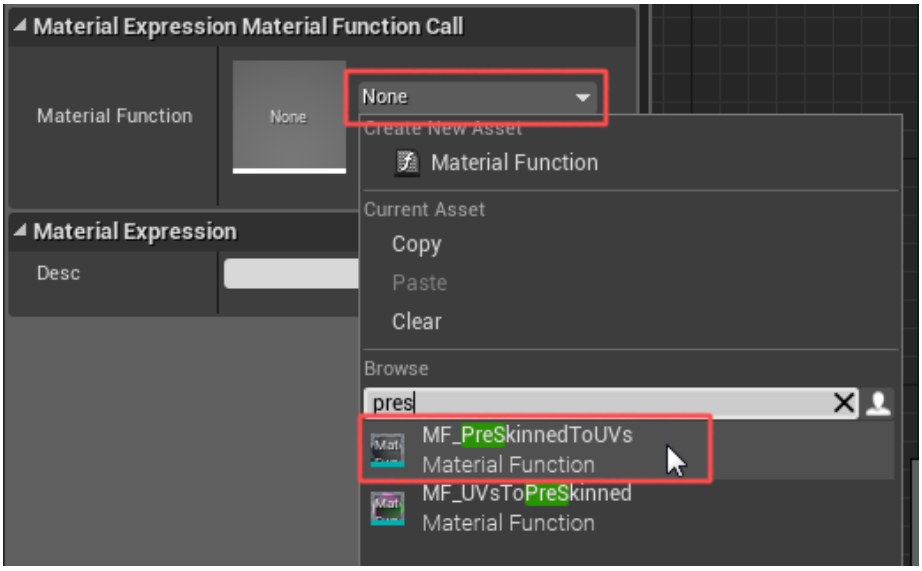
- **BloodSoakingColor:** The color of the blood
- **BloodSoakingRoughness:** The roughness of the blood
- The rest of the properties are just generic material options that you can put your character textures

And if you want to add the blood soaking to your character materials, these are the steps you need to follow:

1. Without having any node selected, in the details panel expand the material section and then set "**Num Customized Uvs**" to **6**, this will add some new parameters at the bottom of your material main node
2. Add two MaterialFunctions:

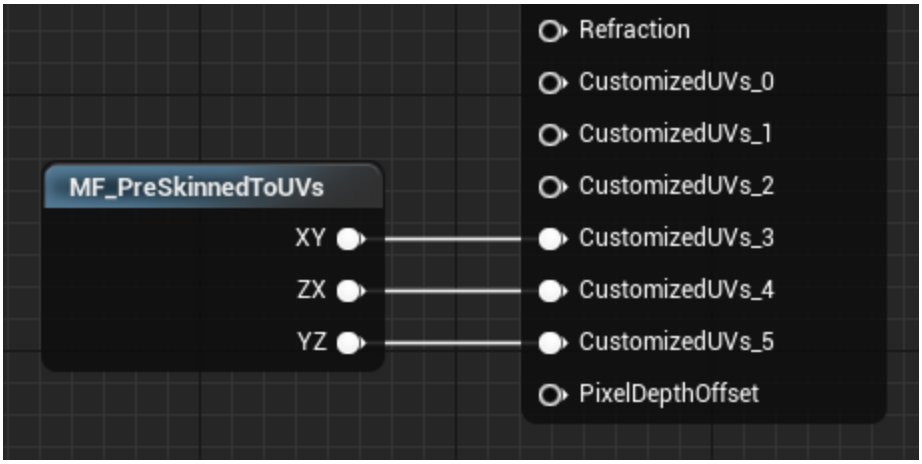


3. In one of them, in the details panel search and select "**MF_PreSkinnedToUVs**":



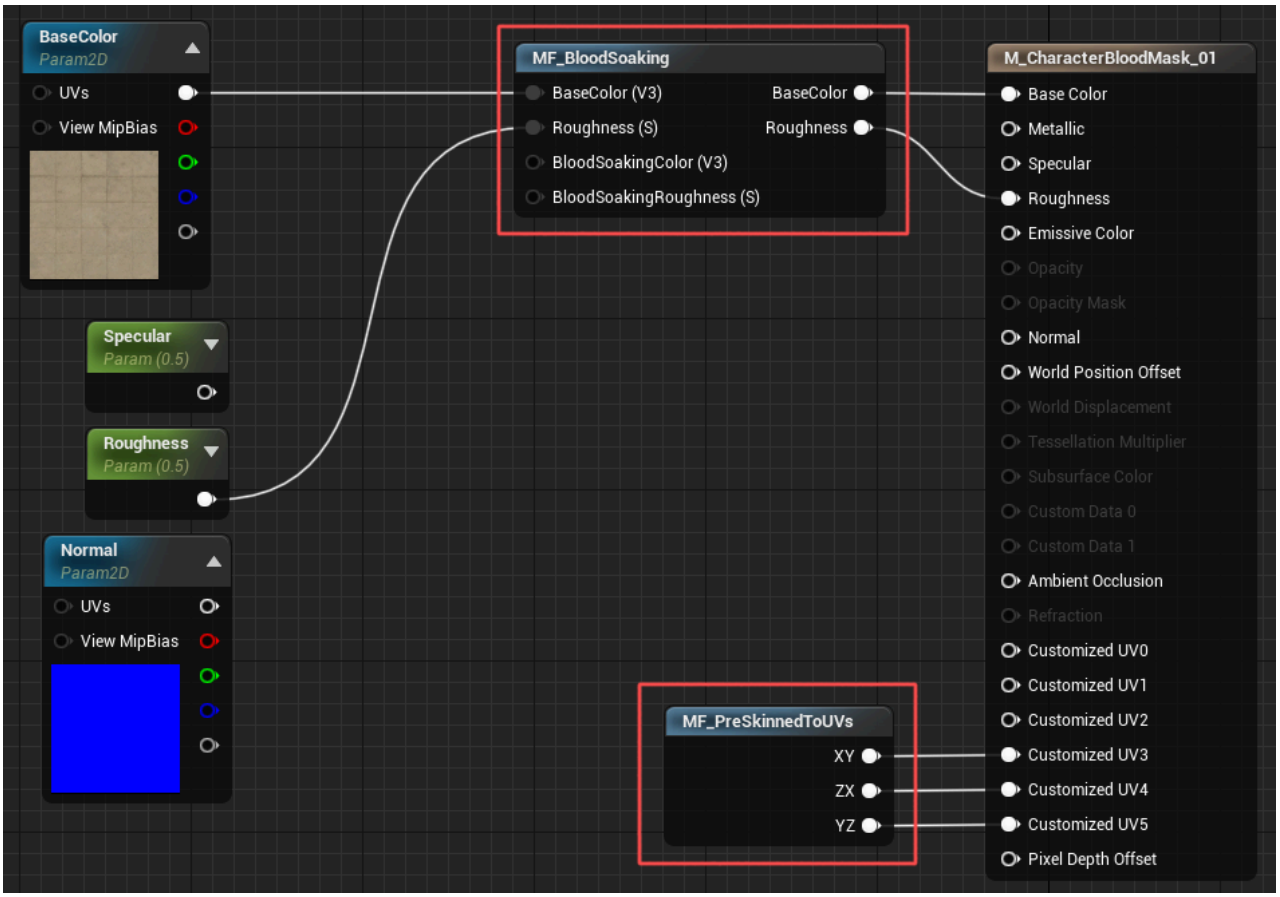
4.

5. Then connect this node to the main material node like this image:

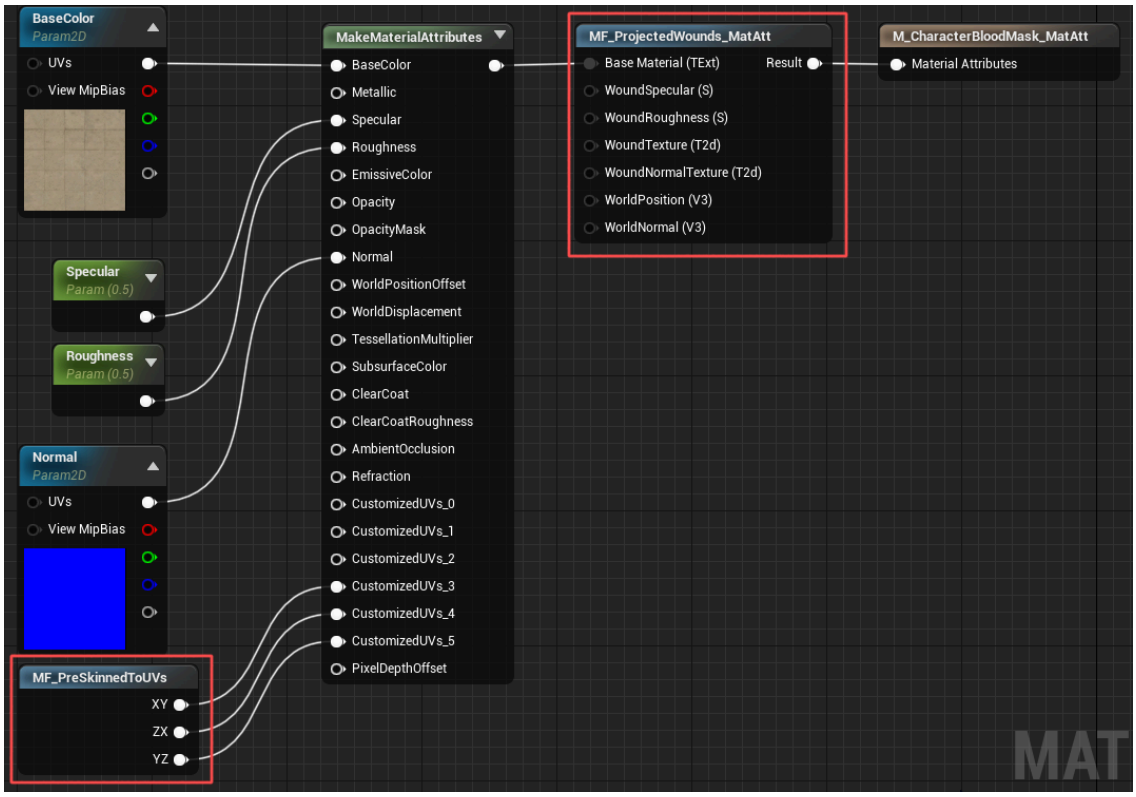


6. And finally there's two MaterialFunctions you can use, depending on the material type you are using:

- **MF_BloodSoaking**: this node modifies your BaseColor and Roughness, by adding a blood layer on top of your materials, and you can add new parameters to the properties you want to change:



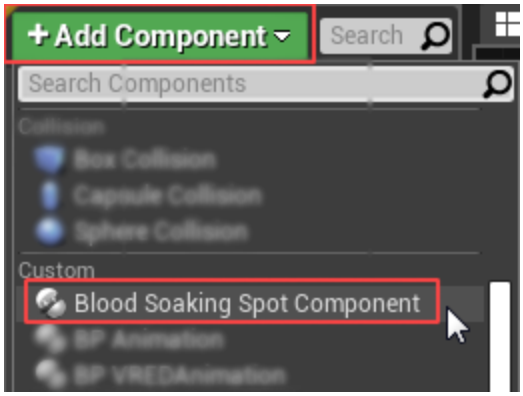
- **MF_BloodSoaking_MatAtt**: similar to MF_BloodSoaking, this one uses MaterialAttributes mode:



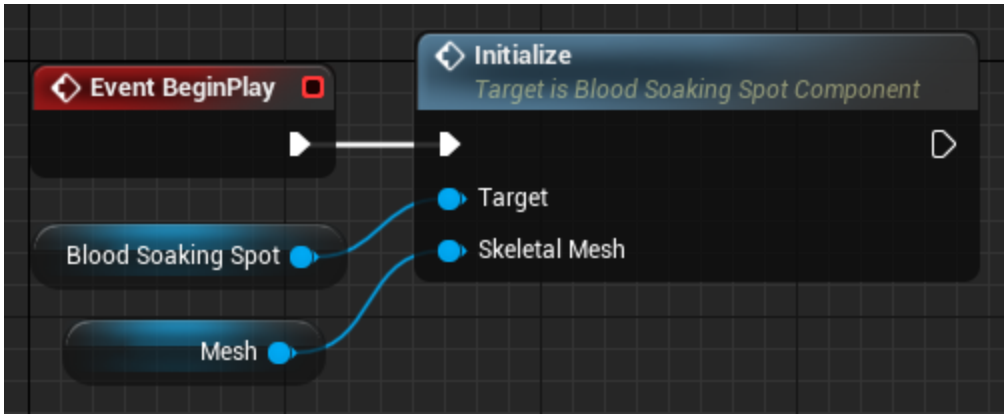
Character Blueprint

These are the steps you need to follow to make your character mesh work in this method:

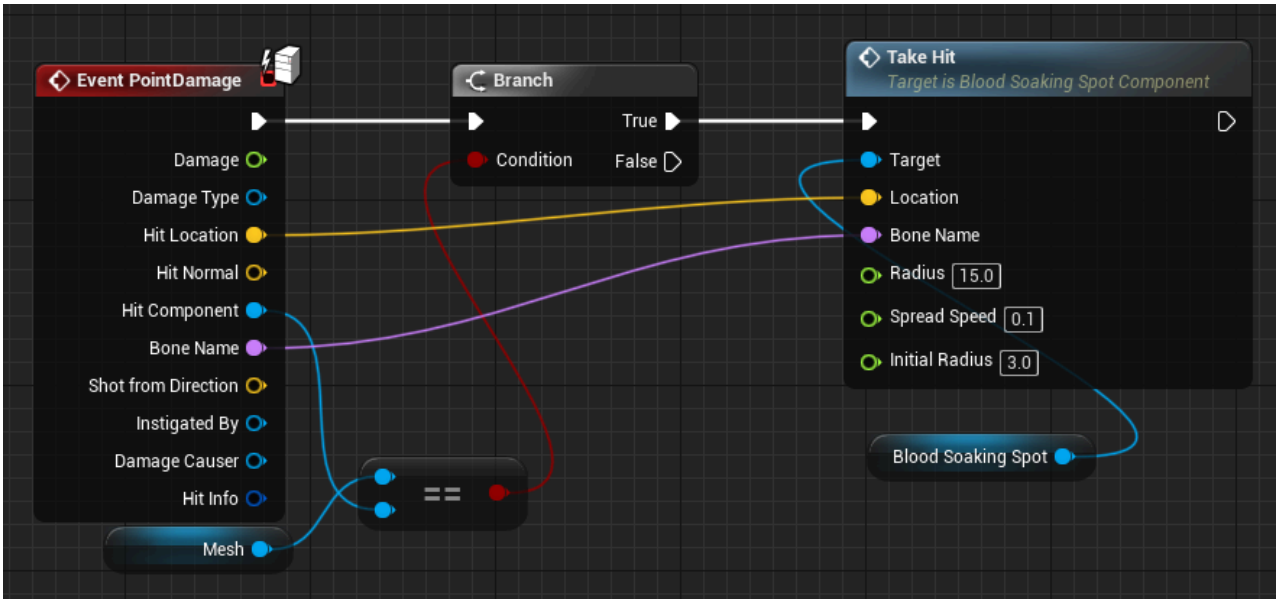
1. Create a Character blueprint, or use any character blueprint you already have
2. Open your character blueprint, and add a new "**BloodSoakingSpotComponent**" component:



3. Find the BeginPlay event and add this to your character BeginPlay:



4. Add a PointDamage to your character and call the TakeHit method from the BloodSoakingSpotComponent:



5. Select the “Mesh” component and in details panel select your mesh

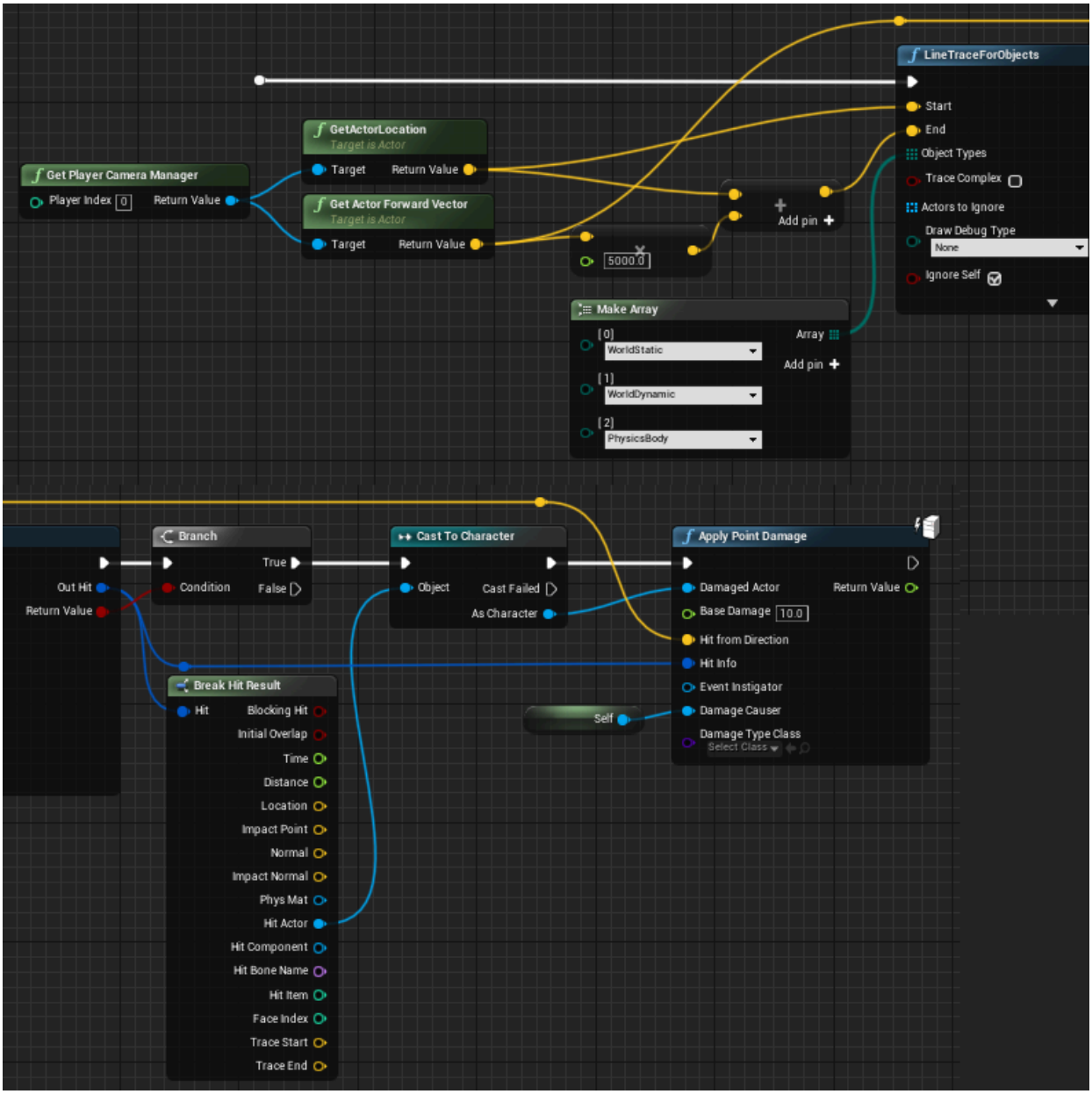
Combined methods

You can also combine the BloodMask and also the projected wounds to the BloodSoaking method, you can find examples of this in the “/Demo/CharacterMaterials” directory.

Integration

Player/Weapon usage

For creating new wounds in your character, all you need to do is, in your weapon or player character, make a line trace using “LineTraceForObjects” using object type “WorldDynamic” for when the character is in the animation state or “PhysicsBody” while is in ragdoll mode, then call the function “Apply Point Damage” using the hit info and direction: (use CH_ExamplePlayerCharacter as a reference)



Extras

F.A.Q.:

BloodMask

Q: How hard is this to integrate to my own project?

A: This system is relatively simple, it uses just a single blueprint for all the character blood mask code. These are the things you need to do: Your character needs a second UV (lightmap UV), a PhysicsAsset and you need to set all of its materials to use a base material that uses the blood mask texture (it's easy when using MaterialInstances), having it done all you need to do is to have a line trace in your weapon and reparent your character to the appropriate character blueprint.

Q: How is the performance cost at runtime?

A: Very minimal, it uses an extra texture in your material and a lerp node to apply the texture over your character texture, besides that it uses two render targets when a new hit is made, but nothing noticeable even with high fire rates.

Q: How much extra memory does it need?

A: Each character creates a render target texture (8bit, red channel only), you can choose the resolution of this texture by just changing a variable.

Q: My character doesn’t show any blood when I hit it, what’s wrong?

A: Make sure that your character mesh has a second UV, that it is using a compatible material, and that you have a PhysicsAsset for your character mesh, and the line trace is hitting your character bones and not the character Capsule component.

ProjectedWound

Q: Q

A: A

BloodSoakingSpot

Q: Q

A: A

Changelog:

> ***Note:*** *Text highlighted in red are updates not yet available (waiting for approval of Epic Games or currently in development)*

Changelog

[1.2.0] - 2020-05-21

Added
- ProjectedWounds method (ProjectedWoundComponent)
- BloodSoaking method (BloodSoakingSpotComponent)
- Option to fix decal glitch on walls using derivatives "DDX/DDY" (thanks David Gutjar)

Changed
- Simplified/organized the player character examples and moved the ragdoll/effects code to WoundSystemExtras
- Changed decal height so it doesn't stretch on walls
- WoundComponent OriginalMaterials renamed to Materials
- WoundComponent added direction parameter to TakeHit
- WoundComponent added projection texture method instead of just a masked sphere (it's a boolean option in the component)
- Fixed blood mask render target having pre-rendered the mesh at the center of the render target at BeginPlay
- Fixed line trace example in player
- Fixed MorphTarget in blood mask render (thanks Art)
- Fixed demo characters calling the hit event while hitting other component
- BloodMaskDebug renamed to M_BloodMaskDebug
- Changed blood decal from translucency to Dbuffer
- Renamed vector parameter in M_BloodRender from Hit to HitLocation
- Fixed demo character mesh arm clipping through clothes

Moved:
- "/CharacterWoundSystem/Meshes" to "/CharacterWoundSystem/Demo/Meshes"
- "CH_ExamplePlayerCharacter" to "./Player"

Removed:
- CH_BloodMaskCharacter as everything can be done using components

[1.1.0] - 2020-04-10

Added
- New method of integration, uses an ActorComponent instead of reparenting the character (see updated documentation)
- Added "MF_BloodMask" and "MF_BloodMask_MatAtt" material functions so it's way easier to add the BloodMask to your own materials (see updated documentation)
- Code to set all needed properties to the character components manually so users don't need to worry about when reparenting blueprints and accidentally losing important settings (Mesh collision set to dynamic)
- WoundRadius parameter added to TakeHit function + renamed "M_BloodRender" material scalar parameter "DamageRadius" to "WoundRadius"

Changed
- Simplified/organized the player character example and moved the ragdoll code to CH_ExampleCharacter (it uses UE4 Character PointDamage event to make things easier)
- Changed decal height so it doesn't stretch on walls

Moved:
- "M_TexturedBloodMask" and "M_SimpleBloodMask" to "/Demo/ExampleAssets/Materials"

[1.0.1] - 2020-04-07

Initial Release