Custom Skin Creation Guide

This guide will explain how to create custom skins (i.e. texture replacements) for characters in *Pawperty Damage*.

1. File Format and Folder Structure

Custom skins are defined using the *JSON* file format: These definition files contain all the information the game needs to load a skin and display it in the game.

The game will look for these definition files in the game's installation directory, at the following path:

```
[Game Directory]/Pawperty Damage_Data/StreamingAssets/Skins
```

As long as your custom skin definitions are somewhere in the *Skins* directory, they can be organized in whatever folders you want:

```
[…]/Skins/CustomSkin1.json

[…]/Skins/My Skins/CustomSkin2.json

[…]/Skins/Foo/Test/HelloWorld.json
```

Organizing your custom skins in folders (i.e. per character) is going to help keep your directory clean and will make it easier to share custom skins, in the future.

> ***Note:*** *Folder names that begin with an underscore (i.e. _Example or _Templates) are ignored.*

2. Custom Skin Definition

As mentioned, custom skins are defined using the *JSON* file format: These files contain information such as…

- What is the name of the skin?
- What character is this skin for?
- What textures does it use?

In the *Skins* directory you will find a folder called *_Example*: It contains a simple example skin to illustrate how everything works.

> *Tip: As a starting point for your own custom skin, simply create a copy of the example folder and rename it to something else (without a leading underscore).*

To edit a *JSON* file, simply open it with your favorite text oder code editor.

2.1 Structure

A simple skin definition in *JSON* looks like this:

```
{
    "Name": "Example Custom Skin",
    "Character": "Rache",
    "MaterialOverrides": [
        {
            "Material": "Body",
            "BaseMap": "Rache_ExampleSkin_C.png"
        },
        {
            "Material": "Hair",
            "Color": "#FF0000"
        }
    ]
}
```

At its most basic level, there are three pieces of data you need to provide:

- ***Name:*** This is the name of your custom skin, as it will be displayed in the game
- ***Character:*** Here you define which character the skin will be applied to (*Rache, Big Mig, Thrash, Vrex*)
- ***MaterialOverrides:*** Array of material overrides, to change the character's default look.

2.2 Material Overrides

To change the appearance of a character in the game, you need to override settings of the character model's *materials*: Each material has a number of different parameters that define its look, such as color tints, image textures, smoothness, etc.

You can selectively override the parameters you want to change: Any parameter you do not override will default to the characters standard look.

For instance, if you only want to change a character's hair color, you only need to override the *Color* parameter of the character's hair material.

The following table shows a list of all parameters a material override can have. Except for *Material*, all parameters are optional:

Parameter	Value	Description
Material	*String*	*Name of the material to override*
Color	*Hex Color*	*Color tint, defined as a hex color, i.e. #FF0000 for red.*
BaseMap	*Texture Path*	*Base color texture*
BumpMap	*Texture Path*	*Normal map (OpenGL format)*
MetallicGlossMap	*Texture Path*	*Texture that combines metallic (RGB) and glossiness (A) maps*
Smoothness	*Number (0.0 - 1.0)*	*Base smoothness/glossiness value: 0.0 is a fully rough material, 1.0 is fully smooth*
OcclusionMap	*Texture Path*	*Ambient occlusion texture*
OcclusionStrength	*Number (0.0 - 1.0)*	*Ambient occlusion amount: 0.0 will show no occlusion, 1.0 will show it at full strength*
Floats	*Float Parameter Overrides*	*Array of generic overrides*
Colors	*Color Parameter Overrides*	*Array of generic overrides*
Textures	*Texture Parameter Overrides*	*Array of generic overrides*

> ***Note:*** *"Floats", "Colors", and "Textures" are advanced options that allow you to modify any shader property of these types. They are arrays of overrides, each of which has a "ParameterName" field to specify which property you want to override, as well as a field called either "Value", "Color" or "Texture" to define what you want to override that property with.*

2.2.1 Hex Color Values

Color value overrides are expressed as “hex” values, the way they are commonly used in web environments.

Color	Hex Code
White	#FFFFFF
Gray	#AAAAAA
Black	#000000
Red	#FF0000
Green	#00FF00
Blue	#0000FF

Graphic programs like *Photoshop* allow you to copy a color as hex value, alternatively there will be many online tools to help pick a color in this format.

2.2.2 Number Values

The overrides for the smoothness and occlusion strength parameters use number values that range from 0.0 to 1.0: Unlike the other basic overrides, these are not strings, and therefore do not use quotation marks:

```
"Smoothness": 0.5,
"OcclusionStrength": 1.0
```

(A value of 0.0 means “0%” and a value of 1.0 translates to “100%”)

2.2.3 Texture Paths

If you want to override the base color map, bump/normal map, etc. with a custom texture, you have to specify a file path that is relative to the skin definition file, pointing to the image texture you want to use:

```
"BaseMap": "SomeTexture.png",
"BumpMap": "Foo/Bar.jpg",
"MetallicGlossMap": "../../Example.tga"
```

> ***Tip:*** *Keeping the textures in the same folder as the definition JSON file will be the easiest option, most of the time.*

3. Textures

Textures for your custom skins should have dimensions of 2048 x 2048 pixels and be saved as either *PNG, JPEG*, or *TGA* files.

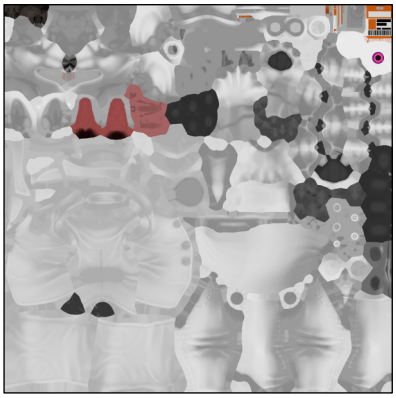
3.1 Templates

To help with custom skin creation, the game provides texture templates that you can use as a starting point.

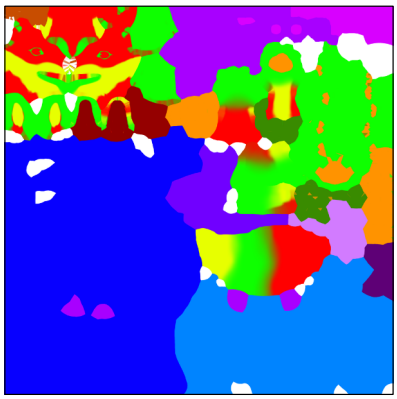
In the game's *Skins* directory you can find a folder called *_Templates*: It contains a *PSD* (*Photoshop*) file for each *Pawperty Damage* character.

Each of these template files has the following layers:

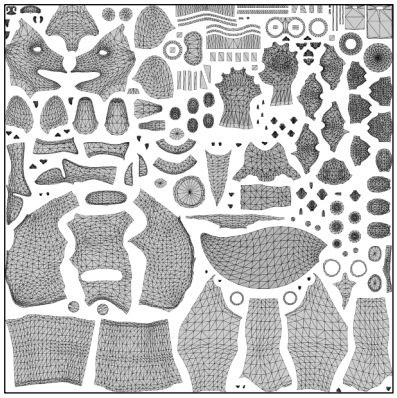
- **Background:** This is a blank skin, that's almost entirely white/gray
- **Skin 1:** The first default skin that comes with the game
- **Skin 2:** The second default skin that comes with the game
- **Color Mask:** A color-coded reference layer that helps you to see what parts of the texture belong together, i.e. all clothes are blue, all fur is red/orange/green, etc. (This color mask has no function in the game and is for reference only.)
- **UV:** The actual UV layout that is used for the texture mapping



Background



Color Mask



UV

> ***Tip:*** *Use the color mask layer to select or mask regions you want to change. You can also use Skin 1 and Skin 2 as references and starting points for your own customized skins.*

4. Seeing your Custom Skin in the Game

If everything is set up correctly, any new custom skin should automatically be loaded when the game is started and show up in the character selection, when cycling through the available skins (by default, *Q* on keyboard or *LB* on an *Xbox* controller.)

5 Troubleshooting

If the custom skin does not show up in the game, something must have gone wrong.

5.1 Common Problems

Here are some common things you can look out for...

- ***JSON* syntax error:** The definition file couldn't be parsed because it is malformed. Make sure it is using correct syntax (i.e. matching parenthesis, commas, etc.)
- **File/folder couldn't be found:** Make sure any custom skin is located in the *Skins* directory and none of its parent folders has a name that begins with an underscore (i.e. *_Example*) as those folders are ignored during import.
- **Character name is incorrect:** Maybe the custom skin is functional, but it could not be associated with a character—make sure the character's name is spelled correctly, in the definition file.
- **Material or parameter names are incorrect:** Make sure material and parameter names are spelled correctly, for any material property you want to override: The names are case-sensitive, use no spaces.
- **Texture paths are invalid:** Double-check the texture paths in the skin definition file and make sure they are relative to the definition file and include the file extension.

5.2 Error Log

Another way to find problems is to check the game's player log, to look for warnings or error messages that give hints as to why a custom skin could not be loaded.

On *Windows*, the log can be found at the following file path:

```
%USERPROFILE%\AppData\LocalLow\Dare Looks\Pawperty Damage\Player.log
```

For *macOS* or *Linux*, refer to Unity's documentation to find the file path.

Search the log for occurrences of the phrase *[PlayerCharacterSkin]* to find any message that was logged by the custom skin system.

Relevant error message might look like this:

```
[PlayerCharacterSkin] No character found with name/ID: Rage

[PlayerCharacterSkin] Invalid skin definition: C:\...
```