Outfit System

An easy to use and powerful outfit management system for Unity.



V 1.0.0

- Incomplete documentation parts will be improved over time.
- Get the most up to date documentation by <u>clicking here</u>.
- Remember you can hover over fields in the "Inspector" window in Unity's editor to read tooltip explanations of each field.
- If you have any questions or need assistance email support at intuitivegamingsolutions@gmail.com.

Table of Contents

- 1. <u>Table Of Contents</u>
- 2. Getting Started
 - 2.a. Importing the Asset
 - 2.b. Included Demo Scenes
 - Make sure to 'Add To Build' the demo scenes before using the 'Catwalk/Closet' UI button(s).
 - 2.c. Included Demo UI
- 3. The OutfitManager Component
- 4. Outfit Users
 - 4.a. The OutfitUser Component
 - Tracks the outfit data for an outfit user even across scenes.
 - 4.b. The OutfitUserPresets Component
 - Allows outfit presets to be specified for an outfit user..
 - 4.c. <u>OutfitUserSO ScriptableObject</u>
 - Contains the outfit data for a user.
 - 4.d. OutfitPreset
 - Holds data to describe an outfit preset.
- 5. Clothing
 - 5.a. The Clothes Component
 - A component that is attached to the root of a piece of clothing.
 - 5.b. The ClothesSlot Enumerate
 - Specifies all 32 possible clothes slots.

- 5.c. The ClothesSlotMask Mask
 - A mask to specify any clothes slots.
- 5.d. ClothesSO
 - Contains information about a specific piece of clothing. Used to track clothing types.
- 5.e. <u>ClothesUtility</u>
 - A static class that provides useful methods for working with clothing, especially with clothing slot masks.
- 5.f. <u>ClothesSlotDictionary</u>
 - A ClothesSlot key'd dictionary with a ClothesSO value that tracks what clothes item is in a given slot.

6. <u>Tattoos</u>

- 6.a. The Tattoo Component
 - A component that is attached to the root of a piece of a tattoo.
- 6.b. The TattooSlot Enumerate
 - Specifies all 32 possible tattoo slots.
- 6.c. The TattooSlotMask Mask
 - A mask to specify any tattoo slots.
- 6.d. TattooSO
 - Contains information about a specific tattoo. Used to track tattoo types.
- 6.e. <u>TattooUtility</u>
 - A static class that provides useful methods for working with tattoos, especially with tattoo slot masks.
- 6.f. TattooSlotDictionary
 - A TattooSlot key'd dictionary with a TattooSO value that tracks what tattoo item is in a given slot.
- 7. Attachments
 - 7.a. The Attachment Component
 - A component that is attached to the root of a piece of an attachment.
 - 7.b. The AttachmentSlot Enumerate
 - Specifies all 32 possible attachment slots.
 - 7.c. The AttachmentSlotMask Mask
 - A mask to specify any attachment slots.
 - 7.d. AttachmentSO
 - Contains information about a specific attachment. Used to track attachment types.
 - 7.e. AttachmentUtility
 - A static class that provides useful methods for working with attachments, especially with attachment slot masks.
 - 7.f. AttachmentSlotDictionary
 - An AttachmentSlot key'd dictionary with an AttachmentSO value that tracks what attachment item is in a given slot.
- 8. <u>Hairstyles</u>
 - 8.a. The Hairstyle Component
 - A component that is attached to the root of a piece of a hairstyle.
 - 8.b. HairstyleSO
 - Contains information about a specific hairstyle. Used to track hairstyle types.
- 9. OutfitData

10. <u>FAQ</u>

Tracks outfit data for hair, all clothes slots, all tattoo slots, and all attachment slots.

NOTE: See 'API Reference.pdf' (<u>online</u>) if you are looking for source code documentation for the 'OutfitSystem' core module.

Getting Started

2.a. Importing the Asset

There are 2 ways to import the 'Outfit System' package.

- a. (Recommended) Using the Unity Editor 'Package Manager'.
 - i. Open the Windows→Package Manager using the Unity editor toolbar.
 - ii. In the upper-left corner of the Package Manager window select 'Packages: My Assets'.
 - iii. Search for 'Outfit System' in the list or use the search bar in the window.
 - iv. Select the asset in the package manager, select 'Download'.
 - v. After the package has finished downloading click 'Import' to import it into the project.
- b. Importing OutfitSystem.unitypackage
 - i. Using the Unity Editor's toolbar select Assets→Import Package
 - ii. In the file explorer that opens navigate to OutfitSystem.unitypackage
 - iii. Double click the package and import it.

2.b. Included Demo Scenes

- *Important* Before using the included demo scenes 'Catwalk' or 'Return To Closet' buttons make sure to add both the closet and catwalk scenes to your projects build using the 'Add To Build' button. This allows the UI to swap between the two scenes.
- **OutfitAndTatooDemo_Closet** a demo scene that allows you to pick between the various hairstyles, clothes, tattoos, and attachments for the demo female.
- **OutfitAndTatooDemo_Catwalk** a demo showing the demo female's outfit state being automatically loaded when switching to a new scene (the catwalk scene) that loads the last saved outfit user state for the demo female and provides a simple showcase scene.

2.c. Included Demo UI

- Table of include demo UI prefabs:

Prefab Name	Description
UI_HairstyleSelection.prefab	A UI prefab for selecting hairstyles.
UI_ClothesSelection.prefab	A UI prefab for selecting clothes that includes a slot selection dropdown, click-to-wear, and click-to-take-off support.
UI_TattooSelection.prefab	A UI prefab for selecting tattoos that includes a slot selection dropdown, click-to-add, and click-to-remove support.
UI_AttachmentSelection.prefab	A UI prefab for selecting attachments that includes a slot selection dropdown, click-to-attach, and click-to-detach support.
UI_ConfirmationPanel.prefab	A simple confirmation panel UI that is used in the demo to request confirmation for actions like delete and overwrite actions.

The OutfitManager Component



A screenshot of the OutfitManager Inspector pane in the Unity Editor for v1.0.0

- The OutfitManager is the main component in 'Outfit System'. It is responsible for defining and managing hairstyles, clothes, tattoos, and accessories for a character.
- <u>Hairstyle</u> components can be referenced by the OutfitManager to define valid hairstyles.
- <u>Clothes</u> components can be referenced by the OutfitManager to define valid clothing.
- <u>Tattoo</u> components can be referenced by the OutfitManager to define valid tattoos.
- Attachment components can be referenced by the OutfitManager to define valid attachments.
- The OutfitManager maintains the current outfit state and manages object visibility for a single outfit manager instance.
- Events:

Event	Description	Arguments
HairstyleChanged	Invoked after the hairstyle has been changed.	Arg0: <u>Hairstyle</u> - The last Hairstyle component or null. Arg1: <u>Hairstyle</u> - The new Hairstyle component or null. Arg2: <u>OutfitManager</u> - The OutfitManager the hairstyle change occurred on.

ClothesChanged	Invoked after clothes in any slot have been changed.	Arg0: <u>ClothesSlot</u> - The ClothesSlot that was changed. Arg1: <u>ClothesSO</u> - The scriptable object reference for the last clothing info, or null. Arg2: <u>ClothesSO</u> - The scriptable object reference for the new clothing info, or null. Arg3: <u>OutfitManager</u> - The OutfitManager the change occurred on.
TattooChanged	Invoked after a tattoo in any slot has been changed.	Arg0: <u>TattooSlot</u> - The TattooSlot that was changed. Arg1: <u>TattooSO</u> - The scriptable object reference for the last tattoo info, or null. Arg2: <u>TattooSO</u> - The scriptable object reference for the new tattoo info, or null. Arg3: <u>OutfitManager</u> - The OutfitManager the change occurred on.
AttachmentChanged	Invoked after an attachment in any slot has been changed.	Arg0: <u>AttachmentSlot</u> - The AttachmentSlot that was changed. Arg1: <u>AttachmentSO</u> - The scriptable object reference for the last attachment info, or null. Arg2: <u>AttachmentSO</u> - The scriptable object reference for the new attachment info, or null. Arg3: <u>OutfitManager</u> - The OutfitManager the change occurred on.

Outfit Users

- The Outfit Users system allows you to easily track outfits, presets, settings, states, and more for an outfit user.
- The Outfit Users system enables you to easily track the current outfit state through scene changes where the character is re-created or already included in the scene.
- The demo includes a tutorial showing how the outfit user system can be leveraged to save the users outfits and custom presets to and load them from a file.

4.a. The OutfitUser Component

Script OutfitUserSo_DemoFemale_00 (Outfit User SO) @ Started () Runtime Only OutfitUserSyncOutfitToUserData Character_DemoFe @ H Character_DemoFe @ Character_DemoFe @ H Character_DemoFe @ Character_DemoFe @ Character_Ch	👍 🗹 Outfit User (Script)		0	t.	:
Settings User © OutfitUserSO_DemoFemale_00 (Outfit User SO) © Started () Runtime Only OutfitUser.SyncOutfitToUserData Character_DemoFe © + - Enabled () List is Empty + - Disabled () List is Empty + - Destroyed () List is Empty + -		DutfitUser			
User OutfitUserSO_DemoFemale_00 (Outfit User SO) @ Started () Runtime Only OutfitUser.SyncOutfitToUserData Character_DemoFe @ + - Enabled () List is Empty + - Disabled () List is Empty + - Destroyed () List is Empty + -	ottinge				
Started () Runtime Only OutfitUser.SyncOutfitToUserData Character_DemoFe + - Enabled () List is Empty List is Empty + - Destroyed () List is Empty + -	lser	ΩutfitUserS0 DemoFemale 00 (Outfit U	ser Si	51	
Runtime Only	Started ()			-/	
Character_DemoFe Character_D	Durations Only				
Character_DemoFe () + - Enabled () List is Empty - Disabled () List is Empty + - Destroyed () List is Empty + -	Runtime Only ▼	Outhtoser.Syncoutht1ooserData		-	
+ - Enabled () - List is Empty + Disabled () - List is Empty + Destroyed () - List is Empty + + -	Character_DemoFe 🕑				
Enabled () List is Empty Disabled () List is Empty + - Destroyed () List is Empty + -					
List is Empty	Enabled ()				
List is Empty					
+ - Disabled () List is Empty + - Destroyed () List is Empty + -					
List is Empty List is Empty + - Destroyed () List is Empty + -					
Disabled () List is Empty Destroyed () List is Empty + -			+		
List is Empty Destroyed () List is Empty + -	Disabled ()				
List is Empty + - Destroyed () List is Empty + -					
+ - Destroyed () List is Empty + -	List is Empty				
Destroyed () List is Empty + -			т.		
Destroyed () List is Empty + -					
	Destroyed ()				
Listis Linpay + -	Liet ie Emntu				
	List is chipty				

A screenshot showing the Inspector pane for the OutfitUser component in the Unity Editor. (v1..0.0)

- The OutfitUser component must be attached to the same GameObject as an <u>OutfitManager</u> component. This component is responsible for tracking the current state of the user's outfit and storing it (or loading it from) a ScriptableObject, specifically a <u>OutfitUserSO</u>, as referenced using the 'OutfitUser.user' field.
- You can leave this component out of your <u>OutfitManager</u> driven character if you do not need any of the state saving or loading features provided by the outfit users system.
- The <u>OutfitUser</u> component implements two extremely important public methods that can be used with *Unity editor events* and/or *C# scripts* to either save or load the relevant <u>OutfitManager</u>'s state to or from the relevant <u>OutfitUserSO</u> ('OutfitUser.user' field):
 - public void SyncOutfitToUserData()
 - Syncs the relevant OutfitManager with data from the relevant OutfitUserSO.
 - This clears the relevant OutfitManager's current outfit (hairstyle, clothes, tattoos, attachments) and attempts to activate all valid outfit items from the items referenced by the relevant OutfitUserSO.
 - public void SyncUserDataToOutfit()
 - Syncs the relevant OutfitUserSO with data from the relevant OutfitManager.
 - This checks hairstyle and iterates through clothes, tattoos, and attachments checking for valid attire and populates the relevant OutfitUserSO so that it matches the outfit currently being worn by the OutfitManager.
- The <u>OutfitUser</u> component also implements *Unity editor events* for common Unity events such as Started, Enabled, Disabled, and Destroyed to allow you to easily configure editor events for situations where you want the simplest set up where you want to invoke

'OutfitUser.SyncOutfitToUserData()' on 'Started (Unity's Start() callback)' as shown in the screenshot above.

4.b. The OutfitUserPresets Component

W	🚱 Outfit User Presets (S	Script)	Ø	랴	:
		OutfitUserPresets			
	Outfits Presets		2		
	=▼ Night				
	Outfit Name	Night			
	Apply Hairstyle				
	Apply Clothing	~			
	Apply Tattoos	V			
	Apply Attachments	~			
	Outfit Data				
	= ► Sports				
			+		

The Inspector pane in the Unity Editor for the OutfitUserPresets component. (v1.0.0)

- The OutfitUserPresets component is attached to the same GameObject as an OutfitUser.
- This component may be left out if you do not need or want preset support for an outfit user (or if you are using an <u>OutfitManager</u> without an <u>OutfitUser</u>).
- The OutfitUserPresets component allows the developer to specify <u>OutfitPreset</u> entries in the editor and/or add or remove them at runtime using the scripting API.
- Each OutfitPreset entry may specify the following:

Field	Description	Туре
Outfit Name	The name of the outfit preset.	string
Apply Hairstyle	When applied does this outfit preset override hairstyle?	bool
Apply Clothing	When applied does this outfit override clothing?	bool
Apply Tattoos	When applied does this outfit override tattoos?	bool
Apply Attachments	When applied does this outfit override attachments?	bool
Outfit Data	The actual <u>OutfitData</u> that tracks what outfit items are in the outfit preset.	<u>OutfitData</u>

4.c. OutfitUserSO ScriptableObject



A partial screenshot of the Inspector pane when selecting an OutfitUserSO. (v1.0.0)

- The OutfitUserSO is a ScriptableObject implementation that contains <u>OutfitData</u> and is intended to contain the current outfit state for an <u>OutfitUser</u>.
- The OutfitUserSO scriptable objects provide a convenient way to visualize and modify the current outfit state for an outfit user from within the editor.

4.d. OutfitPreset

=▼ Night	
Outfit Name	Night
Apply Hairstyle	
Apply Clothing	~
Apply Tattoos	~
Apply Attachme	~
► Outfit Data	

A screenshot showing the property drawer for an OutfitPreset instance in the Unity Editor Inspector pane, outfit data section is collapsed. (v1.0.0)

- An OutfitPreset is a class (data type) that contains a name, type-application settings, and <u>OutfitData</u> that describes the hairstyle, clothes, tattoos, and attachments for an outfit and which elements of an outfit the preset overrides.
- Here is an overview of the fields:

Field	Description	Туре
Outfit Name	The name of the outfit preset.	string
Apply Hairstyle	When applied does this outfit preset override hairstyle?	bool
Apply Clothing	When applied does this outfit override clothing?	bool
Apply Tattoos	When applied does this outfit override tattoos?	bool
Apply Attachments	When applied does this outfit override attachments?	bool
Outfit Data	The actual <u>OutfitData</u> that tracks what outfit items are in the outfit preset.	OutfitData

Clothing

5.a. The Clothes Component

5.b. The ClothesSlot Enumerate

5.c. The ClothesSlotMask Mask

5.d. ClothesSO

5.e. ClothesUtility

5.f. ClothesSlotDictionary

6.a. The Tattoo Component

6.b. The TattooSlot Enumerate

6.c. The TattooSlotMask Mask

6.d. TattooSO

6.e. TattooUtility

6.f. TattooSlotDictionary

Attachments

7.a. The Attachment Component

7.b. The AttachmentSlot Enumerate

7.c. The AttachmentSlotMask Mask

7.d. AttachmentSO

7.e. AttachmentUtility

7.f. AttachmentSlotDictionary

Hairstyles

8.a. The Hairstyle Component

- The Hairstyle component is intended to be attached to any GameObject on a character (even an empty one) and will activate and deactivate said GameObject as said Hairstyle is made the active one, or made inactive respectively.
- The 'Info' field is where the relevant <u>HairstyleSO</u> scriptable object that this Hairstyle implements is specified.

🔻 🧃 🛛 Hairstyle (Script)) 07	:
Script	🖻 Hairstyle	
Settings		
Info	କ୍ତHairstyleSO_LongHair_00 (Hairstyle SO)	

- The 'Active Transforms' setting allows you to reference Transforms that are non-children of the Hairstyle component's GameObject. The Transforms' GameObject will then automatically be activated when this Hairstyle becomes active and deactivated when the Hairstyle becomes inactive.
- The 'Deactivate Transforms' setting allows you to reference Transforms that you want to be automatically deactivated when this Hairstyle becomes active and reactivated when the Hairstyle becomes inactive.

Settings - Activation	
Activate Transforms	0
Deactivate Transforms	0

- The occlusion settings allow you to specify slots for clothes, tattoos, and/or attachments that you want to be automatically hidden while this Hairstyle is active (not unequipped, just hidden).

Settings - Occlusion		
Hide Clothes Slots	Nothing	
Hide Tattoo Slots	Nothing	
Hide Attachment Slots	Nothing	

- The 'Made Active' Unity editor event is invoked whenever the Hairstyle (argument #1) is made active for a given OutfitManager (argument #2).
- The 'Made Inactive' Unity editor event is invoked whenever the Hairstyle (argument #1) is made inactive for the given <u>OutfitManager</u> (argument #2).

8.b. HairstyleSO

- The HairstyleSO scriptable object is responsible for defining the type for a hairstyle.
- Similarly to <u>ClothesSO</u>, <u>TattooSO</u>, and <u>AttachmentSO</u>, the HairstyleSO scriptable object is used to define unique types which will then be referenced by a character-specific <u>Hairstyle</u> component. The HairstyleSO is responsible for allowing a given hairstyle to be identified in a character-independent way.

OutfitData

	None (Hairstyle SO)
Clothes	
	Name (Olathas CO)
неао	None (Clothes SO)
	None (Clothes SO)
	None (Clothes SO)
	None (Clothes SO)
Chauldara	None (Clothes 50)
	None (Clothes SO)
OverShirt	None (Clothes SO)
Chiet	ClothesSO Shirt MiteCrop
UndorShirt	ClothesSO_Shirt_WriteCrop
LoftWriet	None (Clother SO)
DiabtWriet	None (Clothes SO)
LeftHand	None (Clothes SO)
RightHand	None (Clothes SO)
Stomach	None (Clothes SO)
	None (Clothes SO)
Waist	None (Clothes SO)
OverPants	None (Clothes SO)
	None (Clothes SO)
UnderPants	ClothesSO UnderPants Stri
	None (Clothes SO)
	None (Clothes SO)
	None (Clothes SO)
GenitalsLeft	None (Clothes SO)
	t None (Clothes SO)
	None (Clothes SO)
	None (Clothes SO)
	ClothesSO_UnderShoes_So
	None (Clothes SO)
	None (Tattoo SO)
LoftChook	None (Tattoo SO)
	None (Tattoo SO)
RightCheek	
RightCheek	None (Tattoo SO)
RightCheek Chin Neck	None (Tattoo SO) None (Tattoo SO)
RightCheek Chin Neck BackOf <u>Neck</u>	None (Tattoo SO) None (Tattoo SO) None (Tattoo SO)
RightCheek Chin Neck BackOfNeck LeftSho <u>ulder</u>	None (Tattoo SO) None (Tattoo SO) None (Tattoo SO) None (Tattoo SO)

A partial screen capture showing the property drawer for an OutfitData instance as seen in the Unity Editor Inspector pane. (v1.0.0)

- The OutfitData class (data type) contains a reference to the current HairstyleSO that is 'equipped' according to the outfit data.
- The OutfitData class (data type) contains an enumerate key'd dictionary (<u>ClothesSlot</u>, <u>TattooSlot</u>, <u>AttachmentSlot</u>) for clothes, tattoos, and attachments that specify which relevant scriptable objects (<u>ClothesSO</u>, <u>TattooSO</u>, <u>AttachmentSO</u>), if any are in a given slot according to the outfit data.

FAQ

(Frequently Asked Questions)

Q: I am getting an error saying the 'OutfitAndTattooDemo_Closet' and/or

'OutfitAndTattooDemo_Catwalk' scenes have not been added to the build when clicking the 'Return To Closet' or 'Catwalk' UI button(s) in the catwalk or closet demo scenes respectively. **A:** This warning appears when you've forgotten to add the demo scenes to the build settings of your project using 'File \rightarrow Build Settings... \rightarrow Add Open Scenes'.

Simply add the scenes to build settings.

It is recommended when you are done with the demo scenes to remove them from 'Build Settings' if you do not plan to include them on the release build for your project.



A screenshot showing the demo scenes added to the 'Build Settings' of a project.