

So, what is it?

This is an expansion pack for Sci-Fi Ship Controller (SSC is sold separately in the Unity Asset Store). So, you'll first need to own Sci-Fi Ship Controller before you can install and use this expansion pack.

This pack enabled you to perform seamless scene transitions like AAA sci-fi titles! It supports multi-scene jumps and works with the SSC Ship Warp Module.

Table of Contents

SO, WHAT IS IT?.....	1
FEATURES.....	3
HOW DO I GET STARTED?	3
SUPPORT POLICY.....	3
WHAT'S CHANGED?.....	4
OVERVIEW.....	4
HDRP AND URP SCRIPTABLE RENDER PIPELINES.....	4
DEMO JUMPS.....	5
DEMO SEQUENCE 1.....	5
DEMO SEQUENCE 2.....	5
DEMO SEQUENCE 3.....	7
JUMP MANAGER.....	8
BASIC CONFIGURATION STEPS.....	8
JUMP MANAGER – GENERAL SETTINGS.....	8
JUMP MANAGER – JUMPS.....	9
JUMP MANAGER – EVENTS.....	9
JUMP SCENE.....	9
JUMP SCENE - GENERAL TAB.....	9
JUMP SCENE - TRANSITION FROM TAB.....	10
JUMP SCENE - TRANSITION TO TAB.....	10
JUMP SCENE – USAGE SCENARIOS.....	11
JUMP INCLUDE.....	11
JUMP INCLUDE – USAGE SCENARIOS.....	11
JUMP TO HELPER.....	12
JUMP TO HELPER – USAGE SCENARIOS.....	12
INCLUDE HELPER.....	12
INCLUDE HELPER – USAGE SCENARIOS.....	13
PLANET APPROACH.....	13
PLANET APPROACH – GENERAL SETTINGS.....	13
PLANET APPROACH – SHIP SETTINGS.....	14
PLANET APPROACH – FAST TRAVEL SETTINGS.....	14
PLANET APPROACH – EVENT SETTINGS.....	14

PLANET APPROACH – USAGE SCENARIOS	15
PLANET DESCENT	15
PLANET DESCENT – GENERAL SETTINGS.....	15
PLANET DESCENT – DESCENT SETTINGS.....	15
PLANET DESCENT – ASCENT SETTINGS.....	15
PLANET DESCENT – USAGE SCENARIOS	15
COMMON ISSUES	16
COMMON ISSUES – PLANET APPROACH	16
COMMON ISSUES – INCLUDE HELPER	16
COMMON ISSUES – JUMP INCLUDE	16
COMMON ISSUES – JUMP MANAGER AND JUMPS	16
COMMON ISSUES – JUMP SCENE	17
COMMON ISSUES – JUMP TO HELPER	17
COMMON ISSUES – SHIP WARP MODULE	17
RUNTIME AND API	17
RUNTIME GENERAL GUIDANCE.....	17
CHANGING VARIABLE AT RUNTIME	18
DEMO SCRIPTS	18
PLANET APPROACH METHODS - GENERAL	18
PLANET APPROACH METHODS - EVENTS.....	19
PLANET APPROACH METHODS – FAST TRAVEL.....	19
PLANET APPROACH METHODS - STATIC	19
PLANET DESCENT METHODS - GENERAL	19
JUMP INCLUDE METHODS - GENERAL	20
JUMP MANAGER METHODS - EVENTS	20
JUMP MANAGER METHODS - GENERAL.....	20
JUMP MANAGER METHODS - FADE	20
JUMP MANAGER METHODS - SCENES	20
JUMP TO HELPER METHODS - GENERAL	21
JUMP SCENE PROPERTIES	21
JUMP SCENE METHODS – STATIC	21
JUMP SCENE METHODS - EVENTS	22
JUMP SCENE METHODS - GENERAL	22
JUMP SCENE METHODS - TRANSITIONS	22
JUMP SCENE METHODS – STATIC	23
JUMP SCENE PROPERTIES	23
RELEASE HISTORY.....	23
TRADE MARKS	23

Features

- Seamless scene transitions
- Compatible with SSC Warp Module
- Compatible with Ship Camera Module or standard Unity camera
- Approach planet component (BiRP, URP or HDRP)
- Fade in or out a scene
- Move player ship between scenes
- Move ship camera between scenes
- Move gameobjects between scenes
- Call the SSC APIs via Events you can setup in the editor
- Call your own or 3rd party APIs via Events in the editor

How do I get started?

Install Sci-Fi Ship Controller v1.5.2 or newer from the Unity Asset Store. You'll need to be using Unity 2021.3.10 or newer as that is the minimum supported with SSC Seamless.

Then, install SSC Seamless in the project. The content will appear in the Assets\SCSM\SSCXPack2 folder.

If you are using URP or HDRP (rather than Built-in Render Pipeline), you may need to look at the SSCXPack2_SRP_readme.txt in the SCSM\SSCXPack2\SRP folder. This will tell you which SRP package you'll need to install from the SRP folder (otherwise all the materials will be pink).

Open the following scenes from SCSM\SSCXPack2\Demos and add them to Build Settings (if you don't, you'll get the warnings in the Editor Console window and the scene transitions won't work).

- jump1spacescene
- jump1startscene
- jump2startscene
- jump2spacescene
- jump2surfacescene
- jump3startscene
- jump3secondscene

Open jump1startscene or jump2startscene, and using keyboard (W,S keys) and mouse, fly the ship through the jump gate.

The demos will give ideas as to how you could make use of the pack in your own project.

Support Policy

For free support we will investigate reproducible bugs in our code. We may ask you to provide a simple scene with clear instructions on how to reproto the issue. We can provide an upload area for the project files.

If this issue is critical to an announced game release date, we give it high priority. We also help with fleshing out new features that can improve gameplay and that we could add to a new version. In addition, we offer customer support for discovering existing features and how to configure them, both in our [Unity forum](#) or on our [Discord channels](#).

To add "polish" to a game or general help with implementing our products (and even writing custom game-play code) we negotiate a flexible hourly rate which can be time-boxed to fit the studio or indie budget.

For ad-hoc on-going support, and to help us to keep supporting Sci-Fi Ship Controller and expansion packs for a long time, please support us on <https://www.patreon.com/scsmmedia>

We may alter this support policy from time to time without notice.

What's Changed?

Initial Release

Overview

There are two key components, the Jump Manager and the Jump Scene. The first is the overall controller which is placed in your first scene that requires a transition to another scene. You only need one of these per project. The Jump Scene component is placed in each scene that you want to jump from or to.

When transitioning from or to a scene, you have the option to setup an SSC Warp Module (from Sci-Fi Ship Controller v1.5.1+) and have the Jump Manager engage and disengage it for you.

The most common transition, or "jump", includes only 2 scenes. However, it is possible to setup a series of jumps that goes through multiple scenes to get to a destination. An example of this could be a 3-scene sequence where the middle scene only contains an effect. When the effect or warp finishes, the next jump is triggered via an "On Post Disengage" event.

NOTE: You don't have to use the Warp Module during a jump. We use it in our demos to show off several different scenarios that you might not have thought about for your own games.

HDRP and URP Scriptable Render Pipelines

When using HDRP and URP with our demo assets, you will need to apply the SRP packages included in the SCSM\SSCXPack2\SRP folder. This folder contains a readme.txt file which you should check out before proceeding.

Demo Jumps

This pack comes with 3 independent jump sequences to get your started and to help you learn our systems.

Demo Sequence 1

This demo contains jump1startscene and jump1spacescene.

Before you begin, you'll need to load each scene and add them to your build settings. Otherwise, you'll see warnings and the demo won't work.

This demo starts above a city. You fly through the jump gate, which puts you in hyperspace. Shortly after you will emerge in a new star system.

How this demo works:

1. This demo uses a separate ship and ship camera in each scene. This is often all that is required using the same prefabs in each scene (Demo Sequence 2 is more complex and lets you mix and match different scenarios over multiple scenes).
2. The ship starts over a city in jump1startscene with a background of stars created with Space\Celestials.
3. On the JumpManager's "Jumps" tab, there is one jump transversing 2 scenes.
4. In jump1startscene, under Environment the SSCJumpGate2 prefab includes an SSC Ship Proximity component. When the player flies their ship through the gate the ship is detected and the "On Enter Methods" event calls JumpToDestination("JumpGate").
5. The Jump Manager looks through its list of jumps and finds the matching jump by the same name.
6. It looks in the current scene for a JumpScene gameobject which must be a root-level object.
7. It looks for a "Transition From" Ship Warp Module and/or if there are any fade settings.
8. In this case there is no Warp Module and no fading takes place.
9. It starts loading the destination scene (jump1spacescene).
10. When the destination scene is loaded, the Jump Manager enables it.
11. The Jump Manager discovers the JumpScene component in jump1spacescene and checks if there is a "Transition To" Warp Module with or without fading (there is no fade in this scene although there could be!).
12. The Jump Manager engages the warp (SSCWarp1).
13. SSCWarp calls the "On Pre Engage" Warp events which hide the stars (Space->Celestials). Although, a simpler way would be to tick "Is Create Hidden Stars" on the Celestials script. However, this is a demo and we're showing you how to do stuff before the warp runs.
14. The warp effect then runs, after which the "On Post Disengage Warp" run and we show the stars and show two planets.
15. Now we're flying in a different scene with different stars and planets!

Demo Sequence 2

This demo contains jump2startscene, jump2spacescene, and jump2surfacescene.

Before you begin, you'll need to load each scene and add them to your build settings. Otherwise, you'll see warnings and the demo won't work.

This demo adds some additional features on top of what was done in Demo Sequence 1.

This demo starts above a city. You fly through the jump gate and accelerate to lightspeed entering hyperspace. Shortly after you will emerge in a new star system and rapidly decelerate from lightspeed. Then you can fly toward a planet. When you get close, you descend through the thick atmosphere and arrive at the planet surface.

How this demo works:

1. The ship starts in space in jump2startscene with a background of stars created with Space\Celestials.
2. The JumpManager has “Use Fade” enabled on the General tab as scene fading will be used in this sequence.
3. On the JumpManager “Jumps” tab, there are two jumps which will transverse 3 scenes.
4. Any JumpInclude objects may initialise and register with the JumpManager to be transitioned to the next scene. The jump2startscene has a “IncludeFromStartScene” object. This is a dummy gameobject included so you can see how this feature works.
5. In jump2startscene, under Environment the SSCJumpGate2 prefab includes an SSC Ship Proximity component. When the player flies their ship through the gate the ship is detected and the “On Enter Methods” event calls JumpToDestination(“Jump to lightspeed”).
6. The Jump Manager looks through its list of jumps and finds the matching jump by the same name.
7. It looks in the current scene for a JumpScene gameobject which must be a root-level object.
8. It looks for a “Transition From” Ship Warp Module and/or if there are any fade settings.
9. In this case there is a Warp Module and fading takes place during the warp.
10. It starts loading the destination scene (jump2spacescene) and enables the warp (while also doing the fade based on the fade curve from the Jump Scene component).
11. The Jump Manager moves any registered JumpInclude object (e.g., IncludeFromStartScene) out of the source scene (jump2startscene) to a transition scene.
12. Meanwhile, the warp module events are triggered, the environment gameobject is disabled and the warp particle system is started.
13. When the startscene warp has finished, the Jump Manager enables the now loaded destination scene.
14. The Jump Manager discovers the JumpScene component and checks if there is a “Transition To” Warp Module with or without fading (there is no fade in this scene although there could be!).
15. The Jump Manager checks if the JumpScene component in jump2spacescene has “Move Includes” enabled. It does not, so the IncludeFromStartScene gameobject remains in the transition scene and therefore is not used in gameplay while the jump2spacescene is active. This is just an example to show how you can move objects from scene 1 to scene 3 and essentially bypass scene 2.
16. Jump2spacescene also has a JumpInclude files. IncludeFromSpaceScene registers itself with the JumpManager. There is also a JumpInclude on the Ships gameobject which also self-registers with the JumpManager. This has two child objects (a ship and a ship control module). Each of these child objects have a “Include Helper” component which describes the type of object it is, and some other instructional information. These inform the JumpManager, that during the next jump (assuming there will be one), there are some specific gameobjects that will need special attention.
17. The Jump Manager engages the warp (SSCWarp1).
18. In this scene there are no SSCWarp “On Pre Engage Warp” events (but there could be!)
19. The warp effect then runs, after which the On Post Disengage Warp run and we show the stars and show one of the planets.
20. Now we’re flying in a different scene with different stars and a planet, but we’re not done yet!
21. The JumpScene component in jump2spacescene has an “On Post Jump” event.
22. When the jump2spacescene started, the Planet Approach component in the scene was initialised (but not “Enabled” (see the General tab on that component)).
23. The JumpScene “On Post Jump” event starts the planetary approach.
24. Fast travel is enabled, so that starts first, giving the illusion of rapid deceleration from hyperspace.
25. A bit of boost is applied to the ship, then the user can fly the ship toward the planet, still under the control of the Planet Approach component.
26. When the player gets close to the planet, the “On Reached Planet” event commences the next jump. But wait, there is no JumpManager in the current scene, rather it only existing in the transition scene.

That's what the "Jump To Helper" component in the jump2spacescene does. It initiates the "Planet Descent" jump for us.

27. The Jump Scene component in jump2spacescene has no "Ship Warp From" but "Fade Timing" is set to "Post Warp", so the Jump Manager actions the quick fade then loads jump2surfacescene.
28. The Jump Manager also moves the Jump Include gameobjects to the transition scene (remember, IncludeFromStartScene is already there waiting to be told to do something useful).
29. The Jump Manager discovers the "Jump Scene" component in the scene and this time notices the "Transition To" "Move Includes" is enabled. So, it moves all those registered Jump Include gameobject to the destination scene (jump2surfacescene).
30. The Jump Manager also notices that there is a "Ship Warp To" AND a fade that must run for the duration of that warp.
31. The Jump Manager knows some of the child objects have an Include Helper, so it deals with those accordingly. It realises it can populate the Ship Warp Module with the ship and ship camera modules it just moved to the scene, so it does that.
32. The Jump Manager has also discovered that attached to the JumpScene component in the jump2surfacescene is a "Planet Descent" component - did I say there was a LOT going on here?!
33. The Jump Manager engages the SSCWarpDescent1 which include fading using a custom curve while randomly changing the clouds according to the rules from the Planet Descent component.
34. When the planet descent warp completes, the city is revealed (enabled) and the background stars for this scene are shown using the warp's "On Pre Disengage Warp" event.

Demo Sequence 3

This demo contains jump3startscene, and jump3secondscene. These two scenes jump between each other in a loop. In the first scene, you fly through a gate and enter the scene 2. Then after a set period, you return to the first scene.

There are no warps, no fades and no planet approaches. It seems very simple, but there are a couple of features that this demonstrates that the previous two don't.

- You can jump back to a scene that started the sequence
- You can bring your original ship and camera back to that scene
- Message fade in/out during scene transitions

How this demo works:

1. In jump2startscene, the player ship with its attached regular Unity fixed position camera, starts under a disable gameobject. This is important because when we return to the scene, potentially multiple times, we don't want to find an active duplicate of our player ship.
2. On the JumpManager's Events tab, we enable the ship's parent gameobject when the JumpManager begins initialising. This only happens once during our play-through, regardless of how many times we return to "Scene 1".
3. Each scene has a ShipDisplayModule which shows a message indicating which scene we're in.
4. The ship starts in jump3startscene with a background of stars created with Space\Celestials.
5. On the JumpManager's "Jumps" tab, there are two jumps transversing the 2 scenes.
6. Like Demo Sequence 1, under the "Environment" gameobject there is a jump gate. The difference being that here we use a "Jump To Helper" to initiate the jump. Now, this is a bit unusual as the typical advice is to only use "Jump To Helper" in scenes that do not have a JumpManager. But as you no doubt realise, it only has a valid Jump Manager the first time the sequence runs. The second time we visit Scene 1, the Jump Manager is instantly destroyed the moment the scene loads. That is by design as we already have a Jump Manager in our DontDestroyOnLoad transition scene. That's why we need the "Jump To Helper" in this scenario.

7. To avoid the Jump Gate snapping forward when the ship camera is moved the second time Scene 1 is loaded, we'll start the Environment gameobject disabled. We'll enable it the first time using the JumpManager "On Pre Init" event. On subsequent Scene 1 loads, we'll enable it using the JumpScene "On Post Jump" event.
8. So, we go through the gate, and the Jump Manager check the JumpScene "Transition From" details and sees a "On Pre Load Destination" event to hide (and quickly fade out) the "Scene 1" message.
9. There is also a "On Pre Unload Current" event when transitioning from "Scene 1". The event is configured to disable the Environment gameobject to prevent the jump gate from briefly appearing in front of the ship (having just gone through it) as the ship will be repositioned in the destination scene just before the scene with the jump gate is unloaded (see below).
10. However, the now active Ships gameobject has registered itself with the Jump Manager and wants to be transitioned (see "Enable on Init" on the JumpInclude component).
11. There is also a "Include Helper" on the SSCHawk2 ship, plus, one on the child "Camera" gameobject. Typically, you'd only need one on the ship, but in this case, we want the child camera to be updated on the Celestials (background stars) component in each scene.
12. The Jump Manager loads the jump3seconds scene and examining the JumpScene component in that scene realises it should move the Ships gameobject into that scene and take the appropriate action on those "Include Helper" components.
13. The JumpScene in Scene 2 has nothing else of interest EXCEPT an "On Post Jump" event on the "Transition To" tab. The Jump Manager finds the second jump in its list and executes it.
14. But nothing seems to happen. That's because the second jump has a delay of 5 seconds. The player flies around space of a bit and bam, they find themselves back in Scene 1. For astute players they might have noticed the "Scene 2" message does a very quick fade out due to the "On Pre Load Destination" event on the "Transition From" tab in jump3seconds scene.
15. And the player can continue looping between the two scenes for as long as they like.

Jump Manager

This is the core component of the jump system. It needs to be placed in only one scene. This can either be the first scene in your project, or the first scene that requires a jump to another scene.

It contains the global settings for configuring your Jumps. Scene-specific data is held in the JumpScene component in each scene.

The Jump Manager performs and manages the scene transitions. It interacts with the jump scene components in the source and destination scenes. Optionally, it will engage and disengage warp modules in the source and destination scenes, and fire off any event methods.

Basic Configuration Steps

1. In the first scene, a scene, go to GameObject->3D Object->Sci-Fi Ship Controller->SSC Jump Manager
2. In the source and destination scenes, add JumpScene with GameObject->3D Object->Sci-Fi Ship Controller->SSC Jump Scene
3. Configure the Jump Scene components in each scene
4. Add the source and destination scenes to the Unity Build Settings

Jump Manager – General Settings

This is where you'll find all the general global settings for configuring your Jumps.

Property	Description
Initialise on Awake	If enabled, Initialise() will be called as soon as Awake() runs. This should be disabled if you want to control when the module is enabled through code.
Verify Scenes	During testing, it is useful to verify if the scene exists in the build settings.
Use Fade	Fade requires an overlay canvas. This is NOT supported in VR. In the editor, this will automatically add or remove the fade canvas from your jump manager.

Jump Manager – Jumps

This is where you'll find all the jump global settings for configuring your Jumps.

Property	Description
Jump List	
Jump Name	A descriptive name of the jump.
Source Scene Name	The name of the source scene where the jump will transition from.
Dest Scene Name	The name of the destination scene where the jump will transition to.
Jump Delay	The optional time, in seconds, that the jump will be delayed.

Jump Manager – Events

The component event settings.

Property	Description
On Pre Init	These methods get called immediately before the manager starts to initialise. These will act on the scene where the jump manager starts. During gameplay, they should only run once. If this same scene is loaded again after the game has started, these will not run again.

Jump Scene

This is used for in-scene jump configurations. You need one in the source and destination scenes. At runtime, these components are controlled by the Jump Manager.

Jump Scene - General Tab

The jump scene component general settings.

Property	Description
Camera Type	Ignore – used when you want to control the camera outside the Jump Manager. Ship Camera Module – when you have this module with the target being your ship used with the jump. Standard Unity – when you don't have a Ship Camera Module, but you have a Camera object that is looking at, is inside, or following the ship. Typically, you will use this if you have a custom camera setup.
Ship Camera	The ship camera that will be enabled or disabled during a jump.
Camera	The camera used with the jumps in this scene.
Event System	The Event System used in this scene.

Jump Scene - Transition From Tab

The jump scene component settings used in a source scene.

Property	Description
Ship Warp From	The optional Ship Warp Module used when transitioning from this scene to the destination scene.
Engage Warp From	Should the Ship Warp Module be engaged before transitioning from this scene to the destination scene. The scene will transition once the max duration of the warp FX has expired.
Fade Duration	The length of time in seconds, it takes to fully fade in or out the display. Requires Use Fade on Jump Manager.
Fade Timing	When should the fade be initiated?
Preset	Select from one of the preset curves.
Fade Curve	The fade curve used when transitioning from the current scene. When fade curve has a value of 0, the scene will be fully obscured or faded out.
Fade Colour 1	The fade colour when the fade starts.
Fade Colour 2	The fade colour when the fade ends
Colour Curve Preset	Select from one of the preset curves.
Colour Curve	The curve used to blend between the Colour 1 and the Colour 2 when transitioning from the current scene.
Render Mode	The canvas render mode used when fading from a scene. Default: Screen Space – Overlay.
Canvas Position	The position of the canvas when the Render Mode is World Space.
Use Descent	If there is an attached Planet Descent component, should it be used when fading from the scene?
Start Ship Camera	Attempt to start the ship camera when transitioning from the scene. Default: Off
On Pre Load Destination	These methods get called immediately before the destination scene begins to load. This could be useful if you say wish to fade something out of the current scene while the destination is being loaded.
On Pre Unload Current	These methods get called immediately before the current scene is unloaded. Used for transitioning from this scene. These are methods that need to run on the source scene before it is unloaded.

Jump Scene - Transition To Tab

The jump scene component settings used in a destination scene.

Property	Description
Ship Warp To	The optional Ship Warp Module used when transitioning to this destination scene from a source scene.
Engage Warp To	Should the Ship Warp Module be engaged when transitioning to this destination scene from a source scene.
Fade In Duration	The length of time in seconds, it takes to fully fade in the display. Requires Use Fade on Jump Manager.
Fade Timing	When should the fade be initiated?
Preset	Select from one of the preset curves.
Fade Curve	The fade curve used when transitioning to the current scene. When the curve has a value of 1, the scene will be fully visible.
Fade Colour 1	The fade colour when the fade starts.
Fade Colour 2	The fade colour when the fade ends
Colour Curve Preset	Select from one of the preset curves.

Property	Description
Colour Curve	The curve used to blend between the Colour 1 and the Colour 2 when transitioning to this scene.
Render Mode	The canvas render mode used when fading to a scene. Default: Screen Space – Overlay.
Canvas Position	The position of the canvas when the Render Mode is World Space.
Move Includes	Move all the registered JumpInclude object to this scene. They will be automatically unregistered after the move.
Start Ship Camera	Attempt to start the ship camera when transitioning to a destination scene. This may be required if the ship camera in the scene starts in a disabled or uninitialised state. Default: Off
On Pre Unload Previous	These methods get called immediately before the previous (source) scene is unloaded but affect objects in the destination scene. Used for transitioning to this scene. These methods are for things that need to run on the destination scene before the source scene is unloaded or prior to fade and/or warp operations in the destination scene as part of this jump.
On Post Jump	These methods get called immediately after the jump has completed in the destination scene. Used for transitioning to this scene.

Jump Scene – Usage Scenarios

You include one when:

1. This is a source scene for a jump. That is, you want to transition from this scene to another scene. You should configure any applicable settings on the “General” and “Transition From” tabs.
2. This is a destination scene for a jump. That is, you want to transition to this scene from another scene. You should configure any applicable settings on the “General” and “Transition To” tabs.
3. This is both a scene that you jump to, and one you jump from. For example, in Demo Sequence 3, both scenes qualify. You should configure any applicable settings in all tabs.

Jump Include

Add this component to any object that should be included when jumping from one scene to another. The object needs to be a scene root game object due to a Unity limitation.

IMPORTANT:

Do NOT add this to SSCManager, JumpManager, JumpScene, or PlanetApproach objects.

Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the component is enabled through code.
Enable on Init	Include this item in the next jump as soon as the component is initialised.
Register	Automatically register and unregister this component with the Jump Manager.

Jump Include – Usage Scenarios

You could use a Jump Include component to:

1. Keep a player ship active while transitioning between scenes (rather than using identical prefabs in each scene)

2. Keep a player camera active while transitioning between scenes
3. Keep music playing between scenes
4. Move common gameplay scripts and custom components between scenes
5. Move a scoring system between scenes

Jump To Helper

This component that can be added to a scene to help initiate a jump. This is typically NOT required in the scene that contains the (master) JumpManager as you can reference that directly (see Usage Scenarios below). It contains helper API methods called InitiateJump which can be linked to called via Inspector events in Sci-Fi Ship Controller, SSC Seamless, and other 3rd party products. You can also reference it in your own game code and call it directly.

To add one to the scene, go to GameObject->3D Object->Sci-Fi Ship Controller->SSC Jump To Helper.

Jump To Helper – Usage Scenarios

Here are a couple of scenarios where you'll need this in a scene.

1. The most common use case is when you have 2 or more jumps and 2 or more scenes. The Jump Manager should be placed in the first scene that uses a jump or the first scene of your game. During the first jump, the Jump Manager is placed in a transition scene called "DontDestroyOnLoad". At design time, you cannot access this scene as it doesn't exist yet. So, in anything other than the first scene, you can't link to the Jump Manager.
2. You DO have the JumpManager in the scene, but you come back to this scene using another jump. When you return to the first scene, the JumpManager gets removed from the scene because it is a duplicate of what is now in the DontDestroyOnLoad transition scene. In this case you'll need the "Jump To Helper" to let you perform another jump.

So, you add a "Jump To Helper" to your scene and at runtime it talks to the Jump Manager for you.

Include Helper

This component helps you act on an object when it moves to a destination scene. Add this to any object that will be attached to a SSCJumpInclude component or to a child object of that object.

It automatically identifies a Ship or Ship Camera that is attached. However, you can also use it with your own non-SSC gameobjects.

For coders, you can also create your own helper which inherits from this class and override some of the methods.

Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the component is enabled through code.
Item Type	This helps to identify the type of object being moved to a destination scene in a jump.
Reset Pos on Moved	Should the object local position be reset to the original position when it is moved to a destination scene?
Reset Rot on Moved	Should the object local rotation be reset to the original rotation when it is moved to a destination scene?
Update Celestials	If there is a Celestials component in the destination scene, should it be updated with this camera?

Include Helper – Usage Scenarios

Here are some reasons why you might need an Include Helper attached to a gameobject.

1. I want the Jump Manager to know what type of gameobject it is. For example, when ships and ship cameras are moved to a destination scene, we want them to be consistent with how that's done in Sci-Fi Ship Controller. That way, physics calculations etc are performed correctly.
2. I want the local scene SSCManager and SSCRadar to be informed when my player ship is moved to the destination scene. That way, projectiles, effects, beams, destruct objects, and radar items can automatically be dealt with, so I don't have to write custom code for that.
3. A common thing you might want to do is reset the position and/or rotation to the original position or rotation in the originating scene.
4. There is a Celestials component in the destination scene, and I need to set the camera. However, my camera is in a previous scene and is moved to the destination scene using a Jump Include component on it or its parent gameobject.

Planet Approach

This component is used to create the illusion that a ship is approaching a planet from space.

NOTE: This only provides an illusion of planetary approach. It is not designed to allow a ship in a scene to land on the actual gameobject planet. For that, we'd recommend jumping to a planet surface scene, like in the Demo Sequence 2 example. You could build a mesh terrain or use something like Landscape Builder to create your Unity terrain-based scene (which can also convert to a mesh terrain if required).

If you want to approach and land on a spherical planet surface you'd need to either create your own terrain technology with some kind of custom non-popping LOD system (for advanced coders only) or purchase some kind of planet tech like PlanetForge.

Planet Approach – General Settings

The planet approach component general settings.

Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the component is enabled through code.
Enable on Init	Attempt to start approaching the planet as soon as the component is initialised.
Celestials	Component to render stars and distant planets in the sky with built-in Render Pipeline or URP. NOTE: HDRP is not currently supported.
Planet Name	The name of the planet that the ship is approaching.
Planet Radius	The radius of the planet in kilometres.
Start Dist to Planet	The starting distance to the planet in kilometres.
Arrive at Planet Dist	The distance, in kilometres, at which the ship is considered to have reached the planet. This could trigger a planet descent through the atmosphere etc.
Speed Multiplier	This is used to amplify the planet approach speed which helps create the illusion.
Approach Direction	The normalised direction or vector at which the ship will approach the planet. There is a (G)izmos button that can be used to show or hide the direction indicator in the scene view. The (F)ind button selects it in the scene view. You

Property	Description
	can also click on/off the gizmo in the scene view. The Unity Editor Rotate tool changes the direction (or you can type it directly into the inspector). The gizmo will be displayed at the position of the gameobject in the scene view (typically 0,0,0). If in doubt, clear the approach direction to 0,0,0 and it will automatically reset to forwards (0,0,1).
Gizmos Scale	Scale the gizmos in the scene view.

Planet Approach – Ship Settings

The planet approach component general settings.

Property	Description
Approach Ship	The ship that will approach the planet.
Camera Type	The type of camera used with the ship.
Ship Camera	The camera that will be following the approach ship.
Camera	The camera used with the ship when not using the Ship Camera Module.
Limit Pitch/Roll	Enable limit pitch and roll on ship while approaching the planet. Restore original settings after use.

Planet Approach – Fast Travel Settings

The planet approach component fast travel settings.

Property	Description
Fast Approach Duration	The time, in seconds, it takes for the ship to rapidly slow down when first approaching the planet. This could be used when coming out of hyperspace etc.
Fast Travel Dist	The distance, in kilometres, the ship will travel toward the planet from the start distance to planet, before regular approach is engaged.
Preset	Used to quickly setup the Fast Approach Curve.
Fast Approach Curve	The speed curve used when rapidly slowing while approaching the planet. Typically, you will want the curve to start at 1.0 and finish at 0.0 to indicate deceleration when say exiting hyperspace.
Sound FX Set	A set of SoundFX that are randomly selected from when fast travel is engaged.
Sound FX Offset	The local space relative offset from the ship used when instantiating Sound FX.
Add Boost	The amount of boost, in KNewtons, to be applied to the ship when fast travel finishes. This is applied AFTER the “On Finished Fast Approach” event methods (if any) are called.

Planet Approach – Event Settings

The planet approach component events.

Property	Description
On Finished Fast Approach	These methods get called immediately after the ship has finished the fast approach. These are only called if the fast approach duration > 0.
On Reached Planet	These methods get called immediately after the ship has finished approaching the planet.

Planet Approach – Usage Scenarios

Common scenarios include:

1. Give the illusion of a ship rapidly decelerating as it comes out of hyperspace while approaching a planet.
2. Give the illusion of a ship flying toward a planet
3. Triggering a jump to a planet surface scene or in-atmosphere scene when a ship gets close to a planet

Planet Descent

A component, when attached to a SSCJumpScene, supplies data to a jump for descending (or ascending) through a planet's atmosphere. It operates when the jump scene is fading.

Planet Descent – General Settings

Property	Description
Cloud Duration	The range, in seconds, between replacing cloud images during a descent or ascent.
Fade Curve Threshold	From the fade transition curve on the JumpScene component, the maximum the image can be faded in before an image can be replaced.
Cloud Sprites	The list of cloud sprites (textures) used to display during a descent or ascent.

Planet Descent – Descent Settings

Property	Description
Start Altitude	The (fake) altitude the ship will start at when descending to the planet surface.
End Altitude	The (fake) altitude the ship will finish at when descending to the planet surface.
Start Air Speed	The (fake) air speed the ship will start at when descending to the planet surface.
End Air Speed	The (fake) air speed the ship will finish at when descending to the planet surface.

Planet Descent – Ascent Settings

Property	Description
Start Altitude	The (fake) altitude the ship will start at when ascending from the planet surface.
End Altitude	The (fake) altitude the ship will finish at when ascending from the planet surface.
Start Air Speed	The (fake) air speed the ship will start at when ascending from the planet surface.
End Air Speed	The (fake) air speed the ship will finish at when ascending from the planet surface.

Typically, you will probably want to use some kind of warp FX with this feature.

Planet Descent – Usage Scenarios

Common scenarios include:

1. To display something resembling travelling through clouds while descending through a planet atmosphere.
2. To ascend from a planet surface through a cloud layer and into space.

Common Issues

Below is a list of common issues people can encounter that are usually fixed by tweaking the configuration of various components.

Common Issues – Planet Approach

1. When the destination scene is loaded, and a Fade Timing in the Jump Scene is set to “During Warp”, the ship or the ship camera becomes unstable. The cause of this is unknown and is under investigation.
2. My ship flies through the planet. On the General tab, try increasing the “Arrive At Planet Dist”.

Common Issues – Include Helper

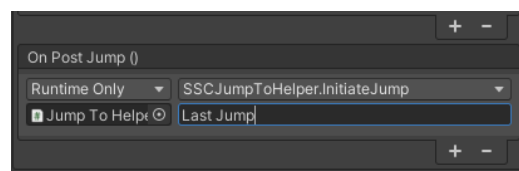
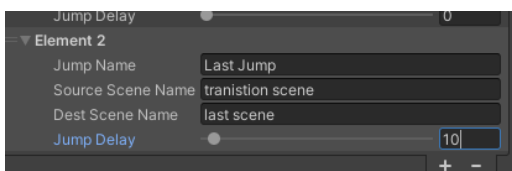
Currently there are no known issues. If you see something, please report it on our Discord channel or contact support.

Common Issues – Jump Include

Currently there are no known issues. If you see something, please report it on our Discord channel or contact support.

Common Issues – Jump Manager and Jumps

1. There are 2 audio listeners in the scene. This can occur on low end hardware or on devices without SSD drives. It can occur because the device cannot activate scenes fast enough and takes one or two extra frames to activate the destination scene. It can also occur due to timing issues that are not consistent – in that they do not happen every time. Please let us know if it adversely affects the overall experience.
2. There are 2 event systems in the scene. Please ensure there is always exactly one event system in the scene. On the General tab of the “Jump Scene” component in the source scene, add the EventSystem from the scene into the “Event System” slot.
3. When I jump to a scene the ship camera is not enabled. Check that there is an SSC Jump Scene gameobject in the root of the destination scene. On the “Transition To” tab, enable “Start Ship Camera”. In the editor at runtime, ensure the “Ship Camera” on the “General” tab is set.
4. How do I initiate a jump when the Jump Manager is not in that scene until it loads at runtime? Add a “Jump To Helper” to the scene by going to GameObject->3D Object->Sci-Fi Ship Controller->SSC Jump To Helper. Now, you can add that object to an event and call the function “SSCJumpToHelper.InitiateJump with the name of your jump from the Jump Manager. Or you could add a public variable JumpToHelper in your own script, drag in the JumpToHelper from the scene and call the function in your own script.
5. I want a transition scene to only be loaded for say 10 seconds, but I don’t want it to contain a Warp Module. How can I do this? In the scene with your Jump Manager, set a delay of 10 seconds for that jump. Then in the transition scene, add a Jump To Helper to the scene and in the “Transition To” tab of the JumpScene, add the JumpToHelper to the On Post Jump event. Set the function to SSCJumpToHelper InitiateJump with a value of “Last Jump”.



6. During a warp in a destination scene, background stars appear behind the warp effect. In the destination scene, on the Celestials component, tick “Is Create Hidden Stars”. When Ship Warp Module’s Events tab, in On Post Disengage Warp, add the Celestials object from the scene to the Runtime Object, and set the Function to “Celestials.ShowStars”

7. Error in HDRP: “Cascade Shadow atlasing has failed, only one directional light can cast shadows at a time”. In the source scene, go to the Jump Scene component. On the “Transition From” tab, add an “On Pre Unload Current” event. Drag the light from the scene into the Object field. Set the “Function” to HAdditionalLightData.EnableShadows(bool). Leave the Boolean value unchecked.

Common Issues – Jump Scene

Currently there are no known issues. If you see something, please report it on our Discord channel or contact support.

Common Issues – Jump To Helper

Currently there are no known issues. If you see something, please report it on our Discord channel or contact support.

Common Issues – Ship Warp Module

1. I cannot connect the Ship to my Warp Module in a destination scene as the ship starts in a previous scene. Add an Include Helper component to your ship. Set the Item Type to “Player Ship”.
2. I cannot connect the Ship Camera Module to my Warp Module in a destination scene as the camera starts in a previous scene. Add an Include Helper component to your Ship Camera Module. Set the Item Type to “Ship Camera”. If the ship camera comes from a previous scene, see the chapter on “Include Helper”.

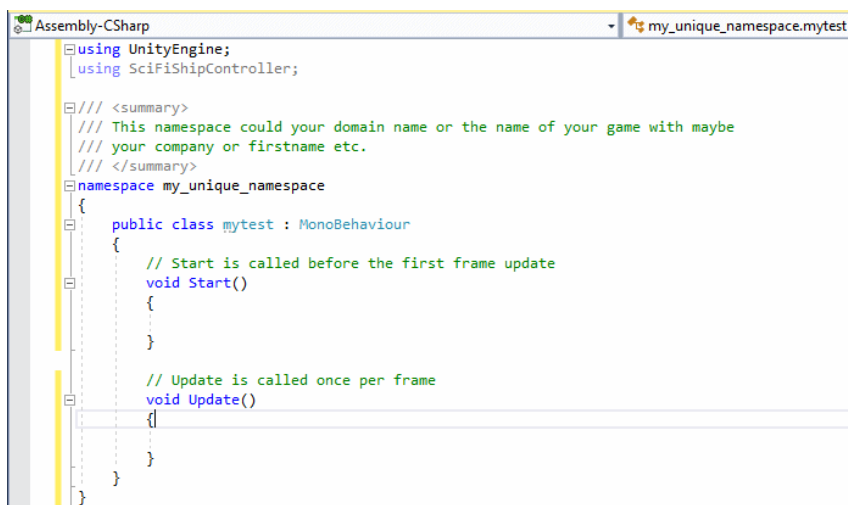
Runtime and API

SSC Expansion Pack 2 is designed to be used in your games. We expect your code to interact with ours. This section, together with the Runtime and API section in the Sci-Fi Ship Controller manual, will help you interact via code with our components.

Runtime General Guidance

Much of our code is well documented and broken down into regions marked with #region #endregion tags. These are expandable in Visual Studio.

When integrating SSC Expansion 2 into your game or project, make sure your scripts are in your own namespace so that they don’t conflict with other people’s code or assets.



```

Assembly-CSharp - my_unique_namespace.mytest
using UnityEngine;
using SciFiShipController;

/// <summary>
/// This namespace could your domain name or the name of your game with maybe
/// your company or first name etc.
/// </summary>
namespace my_unique_namespace
{
    public class mytest : MonoBehaviour
    {
        // Start is called before the first frame update
        void Start()
        {
        }

        // Update is called once per frame
        void Update()
        {
        }
    }
}

```

Public Variables and Properties in our scripts are generally available for you to safely access in your own code. Anything marked “[INTERNAL ONLY]”, “private” or “internal” should never be used in your code as these items are subject to change and will most likely either break your game or make it behave in a strange manner.

Some of our scripts have Public API methods. These are used in our demo scripts and can safely be used in your game code. Look for these Public API regions at the bottom of our scripts.

```
// Sci-Fi Ship Controller. Copyright (c) 2018-2020 SCSM Pty Ltd. All rights reserved.
namespace SciFiShipController
{
    [AddComponentMenu("Sci-Fi Ship Controller/Ship Control Module")]
    [HelpURL("http://scsmmedia.com/media/ssc_manual.pdf")]
    [RequireComponent(typeof(Rigidbody))]
    public class ShipControlModule : MonoBehaviour
    {
        Public Variables
        Private Variables
        Public Static Version Properties
        Public Delegates
        Private Initialise Methods
        Update Methods
        Private Methods
        Events
        Public API Methods - Initialisation
        Public API Methods - Reset, Enable, Disable Ship
        Public API Methods - Enable, Disable Ship Movement
        Public API Methods - Radar
        Public API Methods - Ship Input
        Public API Methods - Docking
        Public API Methods - Ship AI
    }
    Public Structures
}
```

Many of our public variables, properties, delegate call-backs, and methods are documented in the sections below in this manual. Everything else is documented in our script files. Feel free to contact us in our Unity forum or on our dedicated Discord channel if you are unsure of how a variable or method should be used.

Changing Variable at Runtime

Many public variables are modifiable at runtime from within your own code. Variables are commented so that (a) you know what they do, and (b) you can see if they require a method to be called after changing at runtime.

Demo Scripts

From time to time, we may include a collection of helpful scripts that show how certain features can be used in your games or projects. They are subject to change with version upgrades, so are not meant to be used directly in your projects. Instead, the intention is to help you build games with SSC Expansion Pack 2 by providing coding examples. **Do not** make changes to these scripts, instead create your own based on these.

Most scripts have a description at the top and comments throughout.

Script name	Description
Currently none included	

Planet Approach Methods - General

These public methods can be accessed from the SSCPlanetApproach instance in the scene.

Method	Description
ApproachPlanet()	Attempt to start approaching the planet.
GetCelestials ()	Attempt to retrieve the celestials component used with this Planet Approach script.
Initialise()	Attempt to initialise the component.
GetFastApproachDefaultCurve()	Get the default speed animation curve used for fast approaching the planet.
SetApproachDirection (Vector3 newDirection)	Attempt to set the direction the ship will approach the planet.
SetCelestials (Celestials newCelestials)	Attempt to update the celestials field.
SetApproachShip (ShipControlModule newShip)	Attempt to set the ship used to approach the planet.
StopApproachingPlanet()	If already approaching the planet, attempt to abandon the approach.

Planet Approach Methods - Events

These public methods can be accessed from the SSCPlanetApproach instance in the scene.

Method	Description
RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code.

Planet Approach Methods – Fast Travel

These public methods can be accessed from the SSCPlanetApproach instance in the scene.

Method	Description
SetSoundFXOffset (Vector3 newOffset)	Set a new SoundFX offset.
SetSoundFXSet (SSCSoundFXSet newSoundFXSet)	Attempt to up the Sound FX Set for fast travel.

Planet Approach Methods - Static

These public methods can be accessed from SSCPlanetApproach.

Method	Description
CreatePlanetApproach()	Attempt to add a planet approach gameobject and component to the active scene.

Planet Descent Methods - General

These public methods can be accessed from the SSCPlanetDescent component in the scene.

Method	Description
GetImageDuration()	Get a random duration for the next image to be replaced.
GetRandomCloudSprite()	Attempt to return a sprite from the cloud list.

Method	Description
GetCloudSpriteByIndex (int index)	Get the sprite from the list of cloud textures using the zero-based index.

Jump Include Methods - General

These public methods can be accessed from an SSCJumpInclude instance in the scene.

Method	Description
ExcludeFromJumps()	Exclude this object in the next jump
IncludeInJumps()	Attempt to include this object in the next jump
Initialise()	Attempt to initialise this component. Has no effect if already initialised.

Jump Manager Methods - Events

These public methods can be accessed from the static SSCJumpManager at runtime.

Method	Description
RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code. USAGE: SSCJumpManager.RemoveListeners();

Jump Manager Methods - General

These public methods can be accessed from the instance of SSCJumpManager. To get the instance call SSCJumpManager.GetManager() or SSCJumpManager.GetOrCreateManager().

Method	Description
GetJumpByName (string jumpName)	Get the jump using its name. Where possible, use GetJumpByNumber to avoid impacting GC.
GetJumpByNumber (int jumpNumber)	Given the jump number, return either the jump in that position in the list, or null. JumpNumbers begin at 1.
Initialise()	Initialise the jump manager.

Jump Manager Methods - Fade

These public methods can be accessed from the instance of SSCJumpManager. To get the instance call SSCJumpManager.GetManager() or SSCJumpManager.GetOrCreateManager().

Method	Description
GetOrCreateFadeCanvas()	Get or create a new child fade canvas.
SetFadeCanvasSortOrder (int newSortOrder)	Set the sort order in the scene of the fade canvas. Higher values appear on top.

Jump Manager Methods - Scenes

These public methods can be accessed from the instance of SSCJumpManager. To get the instance call SSCJumpManager.GetManager() or SSCJumpManager.GetOrCreateManager().

Method	Description
JumpToDestination (SSCJump jump)	Attempt to jump to a destination scene.
JumpToDestination (string jumpName)	Attempt to jump to a destination scene using the Jump Name. Where possible use JumpToDestination(jumpNumber) or JumpToDestination(jump) by getting the jump first with GetJumpByNumber(..) to avoid impacting GC.
JumpToDestination (int jumpNumber)	Attempt to jump to a destination scene using the Jump Number. Numbers begin at 1.
VerifyScene (string sceneName)	Verify if the scene exists in the Unity Build Settings.
VerifyScenes (SSCJump jump, bool showErrors = true)	Verify if the scenes exist in the Unity Build Settings.

Jump To Helper Methods - General

These public methods can be accessed from any instance of a SSCJumpToHelper in the scene.

Method	Description
InitiateJump (string jumpName)	Execute a jump with the jump name from the master JumpManager which may be in another scene.
InitiateJump (int jumpNumber)	Execute a jump with the jump number from the master JumpManager which may be in another scene. Jump numbers start at 1.

Jump Scene Properties

These public properties are accessible from an instance of SSCJumpScene.

Method	Description
CameraTypeInt	Get the camera type as an int for faster comparison with JumpManager static integers. See also GetCameraType().
IsInitialised	Is the jump scene component initialised?
NonSSCCamera	Get or set the camera used if not using a Ship Camera Module.
ShipCamera	Get or set the ship camera that will be enabled or disabled during jumps.
ShipWarpFrom	Get or set the optional Ship Warp Module used when transitioning from this scene to the destination scene.
ShipWarpTo	Get or set the optional Ship Warp Module used when transitioning to this destination scene from a source scene.

Jump Scene Methods – Static

These public methods can be accessed from SSCJumpScene.

Method	Description
GetManager()	Attempt to get the current jump manager. Typically, it is better to use GetOrCreateManager() except maybe when unloading a scene and you only want to know if the jump manager exists in the project at this point. You might want to use this method in an OnDestroy() method.
GetOrCreateManager()	Attempt to get or create the Jump Manager for the project.
JumpNow (int jumpNumber)	Attempt to jump to a destination scene using the Jump Number. Numbers begin at 1. If you already have a reference to the Jump Manager, instead call JumpToDestination (jumpNumber).

Method	Description
	This is mostly used when you want to make a jump from a warp module or jump scene event that is not located in your first scene with the Jump Manager.

Jump Scene Methods - Events

These public methods can be accessed from any instance of a SSCJumpScene in the scene.

Method	Description
RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code.

Jump Scene Methods - General

These public methods can be accessed from any instance of a SSCJumpScene in the scene.

Method	Description
GetCameraType()	Get the type of camera type used during a jump in this scene.
GetNonSSCCamera()	Get the current non-ssc camera.
Initialise()	This should be automatically called by the Jump Manager.
SetCameraType (SSCJumpManager. SSCJumpCameraType newCameraType)	Set the camera type used during a jump in this scene.
SetEventSystem (EventSystem newEventSystem)	Set the EventSystem used in this scene.
SetNonSSCCamera (Camera newCamera)	Rather than use a ShipCameraModule camera, use a different camera setup.
SetShipCamera (ShipCameraModule newShipCamera)	Sets the ship camera that will be enabled or disabled during a jump.

Jump Scene Methods - Transitions

These public methods can be accessed from any instance of a SSCJumpScene in the scene.

Method	Description
SetFadeFromCanvasPos (Vector3 newPosition)	Set a new fade from canvas position. Used when Render Mode is World Space.
SetFadeToCanvasPos (Vector3 newPosition)	Set a new fade to canvas position. Used when Render Mode is World Space.
SetFadeFromRenderMode (RenderMode newRenderMode)	Set the canvas render mode used when fading from this scene.
SetFadeToRenderMode (RenderMode newRenderMode)	Set the canvas render mode when fading to this scene.
SetShipWarpFrom (ShipWarpModule newWarpFrom)	Set the optional Ship Warp Module used when transitioning from this scene to the destination scene.

Method	Description
SetShipWarpTo (ShipWarpModule newWarpFrom)	Set the optional Ship Warp Module used when transitioning to this destination scene from a source scene.

Jump Scene Methods – Static

These public methods can be accessed from SSCJumpScene.

Method	Description
GetOrCreateJumpScene (int sceneHandle = 0)	Returns the current SSCJumpScene instance for this scene. If one does not already exist, a new one is created. If the SSCJumpScene is not initialised, it will be initialised. For multi-additive scenes, pass in the current scene handle e.g., gameObject.scene.handle.
GetDefaultFadeFromColourCurve()	Get the default animation curve for fading between two colours in source scene.
GetDefaultFadeFromCurve()	Get the default animation curve for fading from a scene.
GetDefaultFadeToColourCurve()	Get the default animation curve for fading between two colours in destination scene.
GetDefaultFadeToCurve()	Get the default animation curve for fading to a scene.

Jump Scene Properties

These public properties can be accessed from any instance of a SSCJumpScene in the scene.

Property	Description
IsInitialised	Is the jump scene component initialised?
ShipCamera	Get or set the ship camera that will be enabled or disabled during jumps.
ShipWarpFrom	Get or set the optional Ship Warp Module used when transitioning from this scene to the destination scene.
ShipWarpTo	Get or set the optional Ship Warp Module used when transitioning to this destination scene from a source scene.

Release History

Initial Release – Dec 2024

Trade Marks

“Unity” is a trademark of Unity Technologies and is in no way associated with SCSM Pty Ltd.

Other names or brands are trademarks of their respective owners.