So, what is it?

This pack enabled you to perform seamless scene transitions like AAA titles! It supports multi-scene jumps and works with or without Landscape Builder and Sticky3D Controller (both assets are sold separately and are available on the Unity Asset Store).

If you already own Sci-Fi Ship Controller (SSC) and wish to seamlessly transition between scenes that use SSC ships, consider SSC Seamless for those scenes. SSC Seamless (aka SSC X Pack2) is also available on the Unity Asset Store.

Loading screens break player immersion, so we want to avoid that. Welcome to LB Seamless. It can also help you move player-related content between scenes at runtime. This avoids having to reconfigure those items (or the player characters themselves), when moving to the next level or scene. In previous scenes, the player could have picked up something or changed configuration as the result of increased XP or level-up.

LB Seamless can also load and unload content as the player moves around a scene. This is accomplished using Addressables and our Batcher feature.

The Downloader feature can check for new content with Unity Cloud Content Delivery and download it as required.

Features

- Seamless scene transitions
- Fade in or out a scene
- Move gameobjects between scenes (including player rigidbody objects)
- Highly configurable to suit your game's requirements
- Call your own or 3rd party APIs via Events in the editor
- Load and unload content within a scene using Addressables
- Download from Unity Cloud Content Delivery system without writing any code
- Works with and support for Sticky3D Controller (sold separately)
- Works with the Unity Character Controller (URP)
- Works with 3rd party Invector Third Person Controller
- Works with Landscape Builder (sold separately)
- Extensive documented APIs that can be called via Events or your own code
- Can be used in the same project as Sci-Fi Ship Controller (sold separately)
- Can be used in the same project as SSC Seamless (sold separately)

Contents

How do I get started?

Install LB Seamless in the project. Restart the editor if prompted (otherwise you'll see compile errors). The content will appear in the Assets\SCSM\LBSeamless folder.

If you are using URP or HDRP (rather than Built-in Render Pipeline), you may need to look at the LBSeamless_SRP_readme.txt in the SCSM\LBSeamless\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\LBSeamless\Demos\Scenes 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.

- LBS_Demo1Scene1
- Runtime\LBS_Demo1Scene2

Open LBS_Demo1Scene1 and run the scene.

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

To quickly setup an automatic jump between your own scenes:

1) In the first (source) scene, go to GameObject->3D Object->LB Seamless->LB Jump Manager
2) On the Jumps tab, add a jump with the first (source) and second (destination) scenes.
3) Set the Jump Delay to 10.
4) Go to GameObject->3D Object->LB Seamless->LB Jump Scene.
5) Add your main camera to the General tab of the Jump Scene.
6) Go to GameObject->3D Object->LB Seamless->LB Jump To Helper
7) Add an empty gameobject and add a LBS Demo Game 1 Start script.
8) Save the scene.
9) Go to File->Build Settings (or Profiles). Add the scene to the list of scenes.
10) Open the second (destination) scene and go to GameObject->3D Object->LB Seamless->LB Jump Scene
11) Save the scene.
12) Go to File->Build Settings (or Profiles). Add the scene to the list of scenes.
13) Open the first scene and run the scene.

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 repro 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 LB Seamless 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?

Version 1.0.1
[NEW] Jump Manager - On Post Init event
[NEW] Jump Manager - GetIncludeHelper() and GetSceneHelper() APIs
[NEW] Scene Helper - On Moved All To Scene events
[NEW] Sticky3D Integration - Set Look state on Moved
[NEW] Sticky3D Integration - scene helper Enable Input on Moved
[CHANGE] Refactor for breaking change in Unity 6.3 of scene.handle
[FIXED] Sticky3D Integration - may lose ref frame if it also was moved
[FIXED] JumpInclude Register may fail

Videos and Tutorials

Name	URL
Trailer 1	https://youtu.be/FbhPaUyYNbo
LB Seamless Basics tutorial	https://youtu.be/74DegR2a76M
Cloud Content Delivery Download tutorial	https://youtu.be/u_KpcYnT0Vk

Overview

There are three key components, the Jump Manager, the Jump Scene, and the Batcher. 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.

Although not essential, if you have a significant amount of content in your scenes, you'll probably need to use the Batcher. This component will help you load and unload content within a scene at runtime with no or minimal code.

Jumps take place between scenes, while Batchers work within a scene. Often, you'll want to combine both.

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.

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\LBSeamless\URP_HDRP folder. This folder contains a readme.txt file which you should check out before proceeding.

Demo Scenes

This pack comes with two independent jump sequences and a choke-point demo to get your started and to help you learn our systems. A Cloud Content Delivery (CCD) template is also included.

Demo Jump Sequence 1

This demo contains LBS_Demo1Scene1 and LB_Demo1Scene2. Four rigidbody bouncing balls are moved between two scenes at runtime. The scenes loop between each other every few seconds.

Demo Jump Sequence 2

This sequence uses Addressable content. It is a revised version of Demo1 with the bouncing balls. The balls, now Addressables, are instantiated by the batcher at runtime.

The first time you open LBS_Demo2Scene1 the “Addressables Group” window may appear. If so, click “Create Addressables Settings”. After the settings have been created, re-open the scene or simply double-click on the scene asset again. This will automatically add the necessary Addressables to the Addressables Groups for this demo.

To build the Addressable content, in the Addressable Groups window, click “Build->New Build->Default Build Script”.

Now you can run LBS_Demo2Scene1 which will loop between LBS_Demo2Scene2 and 3.

Demo Choke-Point Scene

Open the LBS_Demo3Scene. This scene shows how to load and unload Addressable content within a single scene at runtime. You could combine this technique with a floating-point error solution to build large open-world style games.

If you have the “new” Input System in the project, locate the “DemoPlayer1” gameobject in the scene and add the default “Input Action Asset” to the slot provided on the “LBS Demo Player” script.

Run the scene.

Content Downloader Template

Copy this scene (LBS_CCD_Template) to kickstart your development of your own game that uses cloud content. See the “Content Downloader” chapter for full details on how the downloader works.

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 perform fades in the source and/or destination scenes, and fire off any event methods.

Basic Configuration Steps

1. In the first (source) scene, go to GameObject->3D Object->LB Seamless->LB Jump Manager
2. On the Jumps tab, add a jump with the source and destination scenes.
3. In the source and destination scenes, add JumpScene with GameObject->3D Object->LB Seamless->LB Jump Scene
4. Configure the Jump Scene components in each scene
5. If the source and/or destination scenes are standard non-addressable scenes, add them to the scene list in 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 Standard scenes exist in the build settings.
Use Canvas	Fade and ProgressBar requires an overlay canvas. This is NOT supported in VR. In the editor, this will automatically add or remove the canvas from your jump manager.
Canvas Sort Order	The sort order in the scene of the canvas. Higher values appear on top.
Progress Bar	The optional child component under the canvas used to display progress bars during a jump or transition. There is a sample prefab in the Demos\Prefabs\Utils folder.

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	
Scene Name	The name of the source scene where the jump will transition from.
Scene Type	If this is a standard Unity scene or if it is marked as an Addressables object.
Scene Reference	The addressable asset reference for the source scene.
Destination Scene	
Scene Name	The name of the destination scene where the jump will transition to.
Scene Type	If this is a standard Unity scene or if it is marked as an Addressables object.
Scene Reference	The addressable asset reference for the destination scene,
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.
On Post Init	These methods get called immediately after the manager is initialised. 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. Standard Unity – the camera used with the jumps in this scene.
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
FX Type	The type or style of the scene transition effect. If set to Custom, you can subscribe to onInitFXCustomFrom in your own code.
FX Duration	The length of time in seconds, it takes to complete the FX (e.g., to fully fade out the display). Requires Use Canvas on Jump Manager.
Preset	Select from one of the preset curves.
FX Curve	The curve used when transitioning from the current scene. E.g., when curve has a value of 0, the scene will be fully obscured or faded out if FXType is Fade. Typically, you'll want the curve to start at 1.0 and finish at 0.0.
FX Colour 1	The colour when the FX starts.
FX Colour 2	The colour when the FX 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 transitioning from a scene. Default: Screen Space – Overlay.
Canvas Position	The position of the canvas when the Render Mode is World Space.
Start Camera	Attempt to start the camera when transitioning from the scene. Default: Off
Show Progress	Show the progress while loading the destination scene. To setup:

Property	Description
	1) On the Jump Manager "General" tab, enable "Use Canvas" 2) In the scene, add an LBS Progressbar prefab to the Jump Manager canvas 3) Make sure the progressbar UI panel is within the viewable BasePanel area. 4) Drag the Progress bar prefab from the scene into the Jump Manager slot provided. 5) Enable this option on the JumpScene "Transition From" tab in each scene you want the progress bar to appear.
Progress Rate Override	If the rate is greater than zero, it will override the default value in the progress bar in the Jump Manager.
On Pre Jump Delay	These methods get called immediately before a jump delay begins.
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 Activate Destination	These methods get called immediately before the destination scene begins to activate This is after the destination scene is loaded but before Awake(), OnEnable(), and Start() methods run on components in the destination scene.
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
FX Type	The type or style of the scene transition effect. If set to Custom, you can subscribe to onInitFXCustomTo in your own code.
FX Duration	The length of time in seconds, it takes to complete the FX (e.g., fully fade in the display). Requires Use Canvas on Jump Manager.
Preset	Select from one of the preset curves.
FX Curve	The curve used when transitioning to the current scene. E.g., when the curve has a value of 1, the scene will be fully visible if FXType is Fade. Typically, you'll want the curve to start at 0.0 and finish at 1.0.
FX Colour 1	The colour when the FX starts.
FX Colour 2	The colour when the FX 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 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. These are moved BEFORE any Transition From "On Pre Unload Current" and Transition To "On Pre Unload Previous" methods are called.
Stop Source Camera	If there was a camera set in the source scene JumpScene component that is included in the jump, before includes are moved to the destination scene, attempt to stop that camera. Cameras not included, are automatically disabled. Default: Off This could be useful if you want to move a camera to the next scene but don't want to use it immediately, like when a cutscene or other camera animation runs when the destination scene loads.
Start Camera	Attempt to start the camera when transitioning to a destination scene. This may be required if the camera in the scene starts in in a disabled or uninitialised

Property	Description
	state. This happens after On Post Includes Moved and On Pre Unload Prev events but before the previous scene is unloaded. Default: Off
On Pre Includes Move	These methods get called after the destination scene is activated but before any JumpIncludes are moved into the destination scene. It could be used to prepare an environment.
On Post Includes Moved	These methods get called immediately after JumpIncludes have been moved into the destination scene and all IncludeHelper actions have completed. To get a list of the moved items, use LBJumpManager.GetManager().MoveIncludeList.
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 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 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 JumpManager or JumpScene 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.
Initialise Delay	When Initialise() is called, delay for the given time in seconds.
Enable on Init	Include this item in the next jump as soon as the component is initialised.
Register	Register this component with the Jump Manager for inclusion in the next jump.

When the object is registered to be included in a move it will always be moved out of the source scene. However, it will only be moved into a destination scene where “Move Includes” is enabled on the JumpScene component’s “Transition To” tab in that destination scene.

Jump Include – Usage Scenarios

You could use a Jump Include component to:

1. Keep a player 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

Include Helper

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

It can be used with various object types. 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.
Include Helper Key	This can be used to match optional SceneHelpers which can be attached to JumpScene components in each scene.
Disable Input on Moving	If this object is a character controller, the timing (if any) of disabling user input when moving from a scene.
Check Scene Helper	When moved to a new scene, should the destination jump scene be checked for any attached matching SceneHelper components with the same key?
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?
Reset Velo on Moved	If this object has a rigidbody, should the velocity be reset when it is moved to a destination scene?
Set Cam on Moved	If null, should the JumpScene camera property be updated with the attached camera when it is moved to a destination scene?
Enable Input on Moving	If this object is a character controller, the timing (if any) of enabling user input when it has been moved to a destination scene.

Include Helper – Usage Scenarios

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

1. 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.
2. You want to move an object to a specific position in certain scenes. You add an Include Helper to the object and then add SceneHelpers (see below) to the JumpScene component in those scenes.
3. You have a custom action you want to perform on say a player character when it moves to another scene.

Dynamic Helper

Find a root-level gameobject at runtime and add a JumpInclude component. Runs after (most) other Start() component methods in the scene.

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.
Initialise Delay	When Initialise() is called, delay for the given time in seconds.
GameObject Name	The name of the root level object to find in the scene at runtime.
Unhide	If the object is hidden in the hierarchy, unhide it. This can be helpful when debugging a scene.

Scene Helper

This component helps IncludeHelper objects act when moving to a specific scene. It should be attached to a JumpScene component in the scene where it should be actioned.

When the Jump Manager moves an object with a Include Helper to the destination scene, it attempts to find a Scene Helper with the same "Include Helper Key". If it finds a match, it uses the setting on the Scene Helper in at that scene, rather than the default (or global) settings on the Include Helper.

Property	Description
Include Helper Key	The friendly name used by an IncludeHelper to find a matching SceneHelper attached to a JumpScene component.
Reset Pos on Moved	Should the IncludeHelper object local position be reset to the original position when it is moved to a destination scene?
To Position	The world space position where the IncludeHelper object should be moved to in a destination scene.
Reset Rot on Moved	Should the IncludeHelper object local rotation be reset to the original rotation when it is moved to a destination scene?
To Rotation	The world space rotation, in degrees, where the IncludeHelper object should be rotated to in a destination scene.
Reset Velo on Moved	If the IncludeHelper object has a rigidbody, should the velocity be reset when it is moved to a destination scene?
Set Cam on Moved	If null, should the JumpScene camera property be updated with the IncludeHelper camera when it is moved to a destination scene?
DisplayModule	If the IncludeHelper object is a player controller, it may need to connect to a UI display module in a destination scene. If Sticky3D is installed, the Sticky3D Scene Helper can natively use a Sticky Display Module, and/or, if Sci-Fi Ship Controller is installed, a Ship Display Module.
Enable Input on Moved	If this IncludeHelper object is a character controller, the timing (if any) of enabling user input when it has been moved to a destination scene.

LB Seamless includes the following Scene Helpers:

Scene Helper	Description
Generic Scene Helper	Used to set the position and rotation of any moved gameobject.
RBody Scene Helper	For updating a rigidbody that is moved into a scene.
Std Cam Scene Helper	For updating a camera that is moved into a scene.
Sticky3D Scene Helper	For moving a Sticky3D character controller (asset sold separately) into a scene.

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 LB 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->LB Seamless->LB 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.

Content Downloader

You can store addressable content in a Content Delivery Network like Unity's Cloud Content Delivery (CCD) system. This lets you update content rather than having to roll out a new version of the application when content changes. For example, you have updated a prefab or improved a scene's artwork but haven't changed gameplay mechanics. If you game has 50GB of content, you don't want your users to have to download all that content every time you update something.

The LBS Downloader component lets your game check for new content in the cloud and only download what's been updated since the user last played the game.

There is a template scene called "LBS_CCD_Template" in SCSM\LBSeamless\Demos\Scenes which can speed up your development. Simply make a copy and modify as required.

Downloader – General

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.
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 module is enabled through code.
Run on Init	Run a check for new content after initialisation?
Progress Bar	The optional UI progress bar used during downloads.
Status Bar	The optional UI status bar to output the status of the download.
Display Places	Display download size to number of decimal places.

Downloader – Content

This is a list of content to download from the cloud.

Property	Description
Type	The identifier used in this downloadable content.
Label	The content addressables asset label.
Asset Ref	Weak reference to an addressable asset.

Downloader – Events

Property	Description
On New Content	These methods get called when new content is available. This could be used to show a download new content button.
On No New Content	These methods get called if no new content is available. This could be used to enable a "start game" button.
On Post Downloaded	These methods get called immediately after downloads have completed. This could be used to enable a "start game" button after all new content has been downloaded.

Jump and Transition Tips and Tricks

Stalls

If your game stutters or stalls while doing a jump (aka transition), check it in a build before you do anything else. In the Unity Editor, background loading of scenes can still block the main thread of your game.

You'll want to avoid stalls in your game. Most devs don't pay enough attention to what happens when a scene first loads. It is very easy to have a zillion components all calling Awake(), OnEnable() and/or Start() in the very first frame. You've probably seen tutorials where you create many components and then add them to all your prefabs or gameobjects. If they are all "coming online" in the first frame, that is a lot of work for the game engine to do.

If you have a heavy or large destination scene with lots of prefabs (and components) LBS Seamless will load all of those in the background which is unlikely to block the main thread. HOWEVER, activating the destination scene can still cause issues.

Movie and sound files can also be an issue when a scene starts to activate. If the "Load Type" is set to "Decompress on Load", this may take some time. If you have these that must run in first frame (unlikely), try setting the "Load Type" to "Streaming".

Integration with Invector

We have done some integration testing with the third-party asset called Invector Third Person Controller.

> You need Invector TPC v2.6.4c or newer to work with LB Seamless. We don't support the older free version as it was last updated in 2020.

Invector Setup

The vGameController.cs script that comes with Invector has a bug that causes issues with multiple scenes. To fix, open the script and locate the "FindPlayer()" method. Add the following code.

// Patch to fix spawning multiple times
player.onDead.RemoveListener(OnCharacterDead);

```
protected virtual void FindPlayer()
{
    var player = GameObject.FindObjectOfType<vThirdPersonController>();

    if (player)
    {
        currentPlayer = player.gameObject;
        currentController = player;

        // Patch to fix spawning multiple times
        player.onDead.RemoveListener(OnCharacterDead);

        player.onDead.AddListener(OnCharacterDead);
        if (displayInfoInFadeText && vHUDController.instance)
        {
            vHUDController.instance.ShowText("Found player: " + currentPl
        }
    }
    else if (currentPlayer == null && playerPrefab != null && spawnPoint
    {
        SpawnAtPoint(spawnPoint);
    }
}
```

If you are a member of the LB Seamless Beta Program, download the LBSeamless_InvectorTPC_U2022324 unitypackage and add to your project. Add the Demos\Invector\InvectorScene1 and InvectorScene2 to the project Build Settings. Run InvectorScene1.

To create the above two scenes from scratch:

1. Open the Invector_BasicLocomotion scene
2. Save as InvectorScene1.
3. Add a JumpInclude to the vBasicController. Enable “Initialise on Start”, “Enable on Init” and “Register”
4. On the vBasicController, expand “Invector Components” and add another JumpInclude component to ThirdPersonCamera. Enable “Initialise on Start”, “Enable on Init” and “Register”. Set the “Initialise Delay” to 0.1.
5. Rename vBasicController to vBasicController_LBS in the scene and create a new original Prefab of this controller in your Project.
6. Disable the vBasicController_LBS gameobject in the scene.
7. Add the new vBasicController_LBS prefab from the Project panel to “Player Prefab” slot on the “V Game Controller” script attached to the “GameController_Example” gameobject in the scene.
8. Disable the “GameController_Example” gameobject in the scene.
9. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Dynamic Include. Enable “Initialise on Start”, set the “Initialise Delay” to 0.1, and set “GameObject Name” to targetLookAt. Disable the gameobject in the Hierarchy.
10. In the Hierarchy, add another LB Dynamic Include. Set the same options, except this time set “GameObject Name” to Object Container. Disable the gameobject in the Hierarchy.
11. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Jump To Helper.
12. Create a new Empty gameobject in the scene called “StartGame” and add the SCSM\LBSeamless\ Scripts\LB Demo Game 1 Start script. Disable the gameobject.
13. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Jump Scene. Add an SCSM\LBSeamless\ Scripts\Integration\ LBS Invector script to the same gameobject. On the Jump Scene “General” tab, drag the EventSystem from the scene into the slot provided. On the “Transition To” tab, add a “On Post Includes Moved” event. Drag the “LB Jump Scene” gameobject from the scene into the Object slot. Set the Function to LBSInvector.OnMovedIncludes. Add a “On Post Jump” event. Drag in the “LB Jump To Helper” from the scene into the “Object” slot. Set the Function to LBJumpToHelper.InitiateJump with a value of 1. Enable the “Move Includes” property.
14. On both the Transition To and From tabs, set “Fade Timing” to “Fade” and the “Fade Duration” to 0.5.
15. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Jump Manager.
16. On the Jump Manger “General” tab, set “Initialise on Awake”, “Verify Scenes”, and “Use Fade” to on.
17. On the “Jumps” tab, add two jumps. The first “Jump to scene2”, source: InvectorScene1, dest “InvectorScene2”, and Jump Delay say 15 seconds. For the second jump “Jump to scene1”, source: InvectorScene2, dest “InvectorScene1”, and Jump Delay say 15 seconds.
18. On the Jump Manager “Events” tab, add the following gameobjects from the scene to new “On Pre Init” events: “vBasicController_LBS”, “DynamicInclude”, “DynamicInclude” (the two from the scene), “GameController_Example”, and “StartGame”. Set the Functions of all to “GameObject.SetActive” with a value of true. These gameobjects (and scripts) will only be enabled the first time InvectorScene1 is loaded.
19. Save the scene. Add it to the Build Settings.
20. Now make a copy of the scene by Save As InvectorScene2. Add it to the Build Settings.
21. From this second scene, delete the “vBasicController_LBS”, “GameController_Example”, “StartGame”, and both DynamicIncludes.
22. On the LB Jump Scene, “Transition To” tab, on the “On Post Jump” event, set the InitiateJump value to 2.
23. Save InvectorScene2.
24. Open InvectorScene1 and play the scene.

Integration with Sticky3D Controller

You need Sticky3D Controller (S3D) v1.1.9 release or newer to work with LB Seamless. S3D can be purchased from the Unity Asset Store.

S3D characters can be moved between scenes with the aid of the LB Sticky3D Include Helper. Attach this to the player S3D character along with a LB Include Helper. The Include Helper "Item Type" should be set to "Sticky Control Module".

In the destination scenes, you can optionally attach a LB Sticky3D Scene Helper component to the JumpScene gameobject.

Sticky3D Setup

If you are a member of the LB Seamless Beta Program, download the LBSeamless_Sticky3D_U2022324 unitypackage and add to your project. Add the Demos\Sticky3D\S3DPlaygroundScene1 and S3DPlaygroundScene2 to the project Build Settings. Run S3DPlaygroundScene1.

To create the above two scenes from scratch:

1. Open the SCSM\Sticky3DController\Demos\Scenes\ Playground scene
2. Save as S3DPlaygroundScene1
3. Delete disabled gameobject (PlayerBob, Player1, PlayerRod) as we don't need those.
4. Delete S3D_Box1.
5. Expand Environment and delete "Ground" as we don't need that.
6. Add a root empty gameobject to the scene called PlayerObjects. Reset the transform.
7. Move PlayerBryce and ThirdPersonCamera from the scene into the PlayerObjects gameobject.
8. Add a LBJumpInclude component to the PlayerObjects gameobject. Enable "Initialise on Start", "Enable on Init", and "Register".
9. Add a LBIncludeHelper to PlayerBryce. Enable "Intialise on Start" and set the Item Type to "Sticky Control Module". Disable all the "Reset" options.
10. Add a Sticky3D Include Helper to PlayerBryce.
11. Disable the PlayerObjects gameobject
12. Add a root empty gameobject to the scene called NPCs. Reset the transform.
13. Drag PlayerBob_NPC from the scene into the NPCs gameobject.
14. Disable the NPCs gameobject.
15. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Jump To Helper.
16. Create a new Empty gameobject in the scene called "StartGame" and add the SCSM\LBSeamless\Scripts\LB Demo Game 1 Start script. Disable the gameobject.
17. Add a Canvas to the scene, change the Canvas Scaler "UI Scale Mode" to "Scale With Screen Size". Change "Reference Resolution" to 1920x1080. Set "Match" to 0.5.
18. From SCSM\LBSeamless\Prefabs\Utils add the QuitButton and JumpCountdown prefabs to the new Canvas gameobject in the scene.
19. Add a SCSM\LBSeamless\Scripts\LBSCheckInputModule to the EventSystem in the scene.
20. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Jump Scene. On the Jump Scene "General" tab, drag the EventSystem from the scene into the slot provided.
21. On the JumpScene "Transition From" tab, add a "On Pre Jump Delay" event, drag in the "JumpCountDown" gameobject from the scene to the "Object" slot. Set the "Function" to "LBSJumpCountdown.StartCountdown".
22. From SCSM\LBSeamless\Demos\Scripts\Integration\Sticky3D\ folder add LBSS3DDemoPlayground to the JumpScene gameobject.
23. From the scene under "Environment" drag the "Platform" in the "Rotating Platform" slot of the demo script added above.
24. From SCSM\LBSeamless\Prefabs\Props, add Level1Text to the Platform gameobject. Set the RectTransform Pos X: -9.499, Y: 2, Z: -0.24. Set the Rotation to 0,270,0 and the Scale to 3,3,1
25. Disable the "Platform" gameobject in the scene.
26. On the QuitButton, drag the JumpScene from the scene into the "On Click" event Object. Set the "Function" to "LBSS3DDemoPlayground.QuitGame".

27. On the JumpScene “Transition To” tab, add a “On Pre Includes Move” event. Drag in the JumpScene gameobject from the scene into the “Object” slot. Set the “Function” to LBSS3DDemoPlayground. SyncPlayGround.
28. Add a “On Post Includes Moved” event. Drag in the JumpScene gameobject from the scene into the “Object” slot. Set the “Function” to LBSS3DDemoPlayground.OnMovedIncludes.
29. Add a “on Post Jump” event. Drag in the “LB Jump To Helper” from the scene into the “Object” slot. Set the “Function” to “LBJumpToHelper.InitiateJump” and the value to 1.
30. Enable the “Move Includes” property on the “Transition To” tab.
31. In the Hierarchy, click + and 3D Objects->LB Seamless->LB Jump Manager.
32. On the “Jumps” tab, add two jumps. The first “Jump to scene2”, source: S3DPlaygroundScene1, dest “S3DPlaygroundScene2”, and Jump Delay say 15 seconds. For the second jump “Jump to scene1”, source: S3DPlaygroundScene2, dest “S3DPlaygroundScene1”, and Jump Delay say 15 seconds.
25. On the Jump Manager “Events” tab, add the following gameobjects from the scene to new “On Pre Init” events in this order: “Environment\Platform”, “PlayerObjects”, “NPCs”, and “StartGame”. Set the Functions of all to “GameObject.SetActive” with a value of true. These gameobjects (and scripts) will only be enabled the first time S3DPlaygroundScene1 is loaded.
26. In the Hierarchy, click + and 3D Objects->Sticky3D Controller
27. Save the scene. Add it to the Build Settings. Generate Lighting.
28. Now make a copy of the scene by Save As S3DPlaygroundScene2. Add it to the Build Settings.
33. From this second scene, delete the PlayerObjects, StartGame, and JumpManager.
34. On the LB Jump Scene, “Transition To” tab, on the “On Post Jump” event, set the InitiateJump value to 2.
35. Under Environment, Platform, locate Level1Text and change the TextMeshPro – Text from “Level 1” to “Level 2”.
36. Under Platform, locate “Floor” and drop the SCSM\LBSeamless\Demos\Materials LBSPastelA3 on it to make it a little easier to determine that we’ve changed scenes.
37. Save S3DPlaygroundScene2. Generate Lighting.
38. Open S3DPlaygroundScene1 and play the scene.

Sticky3D Addressable

You may have a player Sticky3D character that you wish to move between Addressable scenes. When objects move between Addressable scenes using the JumpInclude component, they may lose their materials and/or textures. This is a known issue with Unity Addressables. See the “Working with Addressables” chapter.

To resolve this, the Sticky3D character prefab must be marked as “Addressable” and instantiated into a scene. Here are the abbreviated steps required.

1. In the first non-addressable scene, place a camera and add a Jump Include to that camera. Tick “Initialise on Start”, “Enable on Init” and “Register”. Gameobject Tag to “MainCamera”.
2. Mark your other scenes in your game “Addressable” and move them into their own groups.
3. Back in the first scene, add a Jump Manager and setup the jumps required.
4. Add a Jump Scene and add the camera from the scene on the “General” tab.
5. In the second scene (your first jump), add a Jump Scene component.
6. If your character can switch to 3rd person, add an empty gameoject and reset the transform. Rename it “S3DPlayer”. Add a Jump Include component. Tick “Initialise on Start”, “Enable on Init” and “Register”.
7. Add your Sticky3D Controller character prefab and third person camera as children of that S3DPlayer.
8. On the child Sticky3D character, add an Include Helper. Enable “Initialise on Start” and “Check Scene Helper”. Set the “Item Type” to “Sticky Control Module”. Remember the “Include Helper Key” (which you can change if required).
9. Also add a LB Sticky3D Include Helper to the Sticky3D character.
10. Create a new prefab from the S3DPlayer gameobject and delete the one in the scene.
11. Mark the S3DPlayer prefab “Addressable” and move it into its own group.
12. Still in the second scene, add a Jump To Helper

13. Still in the second scene, add an LBS Batcher. Disable “Initialise on Awake”, “Initialise on Start”, and “Is Run on Init”.
14. Add a batch and name it “Add S3D Player”
15. Enable “Is Batch Enabled” and “Is Pre Init”
16. Add an “On Pre Batch” event add drag the “Jump To Helper” into the Object slot. Set the Function to “LBJumpToHelper.StopMainCamera”. Tick the option to also disable the audio listener.
17. Add an item and set the type to “Initiate Addressable”
18. Add the S3DPlayer prefab we created above.
19. Disable the LBS Batcher gameobject.
20. Add an LBS RunOnce component to the scene
21. Add a “Run On Start Once” event.
22. Add the Batcher from the scene to the Object slot. Set the Function to “GameObject.SetActive” and tick the true box.
23. Add another Event. Add the Batcher and set the Function to “LBSBatcher.Initialise”
24. Add another Event. Add the Batcher and set the Function to “LBSBatcher.RunBatchesAll”
25. Optionally, in each scene the S3D character moves to, on the JumpScene component in those scenes, add a “LB Sticky3D Scene Helper” component. Use the “Include Helper Key” we used earlier. Enable “Set Cam On Moved” and “Set Ref. Frame”. In the “Ref. Frame” field add the initial reference frame for the character being careful to select on with a scale of 1,1,1.
26. Optionally on the “LB Sticky3D Scene Helper” component you could enable “Reset Velo on Moved” and set a different position and/or rotation for each scene.

Proximity Component

This component let you call your game code, many Seamless API methods, and/or set properties on gameobjects when an object with a collider enters or exits an area of your scene. Essentially, it saves you time from having to write collider trigger code. There is also an option to check the object’s Unity tag.

If no tags are provided, all objects can affect this area. NOTE: All tags MUST exist.

A typical use case is when a character enters an area, and you want to perform some kind of custom task or make something happen in your game. It could be something as simple as turning on/off a light as the character enters or exits a room.

You could also use it with the Batcher to instantiate or release addressable prefabs.

To add one to the scene, use the 3D Object -> LB Seamless menu to create a new gameobject with either a sphere or box trigger collider.

Scene Looping Secrets

How can I move between scenes while allowing the player to return to the original scene? In the demos, you might have seen that LB Seamless can help you build looping scenarios.

The key elements are:

- Add the player and camera as children of a root-level gameobject in the first scene. E.g. PlayerObjects
- Add a LBJumpInclude to the PlayerObjects gameobject.
- Start with PlayerObjects disabled.
- Enable the PlayerObjects the first time using the JumpManager “On Pre Init” events. Alternatively, you could use the LBS RunOnce component.
- In each scene, add a LBJumpScene as root level gameobject (tick “Initialise on Start”, “Enable on Init”, and “Register”.

- You almost always want a LBJumpToHelper in each scene that you can call from your own code or using an event on a component.

If you wish to always jump after a period, do the following:

- On the Jump Manager, for each jump that you wish to automatically jump after loading the destination scene, set the "Jump Delay" time in seconds.
- On the LBJumpScene in each scene, on the "Transition To" tab, add a "On Post Jump" event. Drag the "LB Jump To Helper" from the scene into the Object slot. Set the Function to "LBJumpToHelper.InitiateJump" and enter the Jump number from Jump Manager.

Working with Addressables

Addressables can help you break scenes up into smaller pieces and can also be used to automatically deliver and update game content from the internet. For example, new content could be loaded from Unity's Content Delivery Network.

Another advantage of using Addressables, is that scenes can be made smaller, which means faster load and activation times. The scene can start with the minimum content required, and content can be loaded (and unloaded) as required.

The Jump Manager can load Addressable scenes. Typically, you'll want your first scene to be very minimalistic and non-addressable. The first scene could be a splash-screen or maybe a menu scene that gets users into the game.

If moving objects between addressable scenes with JumpInclude, instantiate them into the game as addressables. If you don't do this, when they move to another scene, the materials and/or textures for those objects may be unloaded with the source scene. This is a known issue with Unity Addressables. You can use the LBS Batcher and LBS RunOnce components to help with this task.

The LBS Batcher can be used to spawn regular gameobjects or addressables.

Batcher

This component is used to perform actions in groups or "batches". Each batch contains one or more items. The action perform on the item is dependent on the Item Type. For example, activate a gameobject or instantiate a prefab.

The Batcher can be used to spawn regular prefabs or addressables. You can call Batcher APIs from the LBS RunOnce component. E.g. Initailise() and/or RunBatchesAll().

If the addressable prefabs are stored in the cloud, see also the "Content Downloader" chapter.

Batcher – General Properties

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.
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 module is enabled through code.
Run on Init	Should all batches be run immediately after initialisation?
Start Delay	The time, in seconds, to delay running the first batch after initialisation.
Batches	The list of batches.

Batcher – Batch Properties

Property	Description
Batch Name	Descriptive name for the batch.
Batch Duration	The total time, in seconds, that the items will be attempted to be deployed.
Start Delay	The time, in seconds, before this batch begins.
On Pre Batch	These methods get called immediately before the batch is executed.
On Post Batch	These methods get called after the batch has been completed.
Item List	The list of items to action for this batch

Batcher – Item Properties

Property	Description
Item Type	The type of action or operation to be performed on this item. ActivateGameObject – the object will be activated DeactivateGameObject – the object will be deactivated InstantiateAddressable – the addressable will be instantiated InstantiatePrefab – the prefab will be instantiated
GameObect	A gameobject in a scene that can be activated or deactivated.
Prefab	A prefab that can be instantiated.
Prefab Ref	The addressable asset reference for the prefab.
Keep Ref.	If this is an instantiated prefab, keep the gameobject instance (if any) that has been instantiated. WARNING: Do not enable if you wish to destroy the object outside the batcher. Instead, if this is an addressable prefab, attach a LBSDestroy component to the prefab.
Position	The instantiate world space position.
Rotation	The instantiate world space rotation.
Parent	The optional instantiate parent object.

Cloud Content Delivery

To use Unity's Cloud Content Delivery your project must be linked to Unity Cloud. This can be done when a project is created with the Unity Hub or by clicking on the three dots next to a closed project in the Hub.

From within the Unity Dashboard (https://cloud.unity.com/) locate your project and create an environment, a bucket, and a badge using the Unity CCD instructions.

The badges will basically be your versions or releases of content (e.g. Production for online users or Latest for internal testers).

NOTE: Release management of Addressable content is outside the scope of this document. See the Unity documentation or online videos from games studios that discuss using CCD and CDN systems.

In the Addressables Profile, ensure that Remote is using Cloud Content Delivery (either Automatic or with a specific Environment, Bucket, and Badge.

For CCD content groups you'll want to have a schema type of "Content Packing & Loading". The "Build & Load Paths" should be set to "Remote". You will probably want to set "Bundle Mode" to "Pack Separately" so that if you change some content, the users only need to download what has changed.

Common Issues

Common Issues – Jumps

Currently there are no known issues. If you see anything, please let us know on our Discord channel.

Common Issues – ProgressBar

1) I can't see the progress bar during transitions. At design time, in the scene view, make sure the progress bar is within the visible area under JumpManager, Canvas, Base Panel. On the Jump Manager "General" tab, ensure "Use Canvas" is enabled and the Progressbar from the scene is set. On the JumpScene in scenes you want to see the progressbar, on the "Transition From" tab, ensure "Show Progress" is enabled.

Common Issues – Sticky3D characters

These assume you have the Sticky3D Controller asset (sold separately) is installed in your project.

1) How do I set the Reference Frame for my character when moving into a destination scene? Add an LB Include Helper and LB Sticky3D Include Helper to the character's Sticky Control Module. Set the "Item Type" to "Sticky Control Module". Set the "Include Helper Key" to something unique or say "S3D Player". Enable "Check Scene Helper". In the destination scene, on the JumpScene gameobject, add an LB Sticky3D Scene Helper. Set the "Include Helper Key" to match the one set on the Include Helper. Enable "Set Ref. Frame" and drag the gameobject from the destination scene into the slot provided.
2) How do I use a Sticky Display Module in a destination scene for my character that comes from another scene? Setup LB Include Helper, LB Sticky3D Include Helper, and LB Sticky3D Scene Helper components as described in #1 above (ignore the Ref. Frame setup unless you want that too). In the destination scene, drag the Sticky Display Module from that scene into the LB Sticky3D Scene Helper "Display Module" slot.
3) How do I set the position and/or rotation of a character when moving to a new scene? See #1 and enable "Set Pos on Moved" and/or "Set Rot on Moved" in the scene helper instead of, or in addition to the Ref. Frame.

Common Issues – Transition FX

1) The default FX Curve works with a Wipe, but my curve seems to make the wipe go in the wrong direction. For "From" transitions, ensure the curve starts at 1.0 and ends a 0.0. For "To" transitions, ensure the curve starts at 0.0 and finishes at 1.0

Runtime and API

LB Seamless is designed to be used in your games. We expect your code to interact with ours. This section 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 LB Seamless 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.

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 is 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.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

// Landscape Builder Copyright (c) 2016-2025 SCSM Pty Ltd. All rights reserved.
namespace LandscapeBuilder
{
    /// <summary>
    /// Batch Objects in groups - might want to enable multiple in same frame or even have dependencies
    /// </summary>
    [HelpURL("https://scsmmedia.com/lbs-documentation")]
    public class LBSBatcher : MonoBehaviour
    {
        Public Variables

        Public Properties

        Protected Variables

        Protected and Public Variables - Editor

        Initialisation Methods

        Protected Methods

        Public API Methods

        Public Static Methods
    }
}
```

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 LB Seamless 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
LBSDemoBall1	Simple demo script to use with a bouncing ball in LB Seamless.
LBSDemoDeployBalls	On start this will instantiate addressable balls in the demo scene.
LBSDemoGame1Manager	Simple demo script to manage a game. It should be attached to the jump manager and will therefore be always present at runtime.
LBSDemoGame1Scene1	Simple demo script that detects when items are moved into a scene.
LBSDemoGame1Scene2	Simple demo script that detects when items are moved into a scene.
LBSHideCursor	Sample script hide/show the mouse pointer due to mouse (in)activity. Drop it onto a gameobject in the scene.
LBSJumpCountDown	Demo script to show countdown of a delayed jump.
LBSSampleChokePoint	Simple sample script shows how you could load and unload content using addressables.
LBSSampleGetSourceCamera	Simple sample script shows you how to get the camera from the source scene during a jump.

Batcher Methods - General

These public methods can be accessed from the instance of LBSBatcher.

Method	Description
AddBatch (LBSBatch newBatch)	Attempt to add a batch to the list of batches.
DestroyBatchItems (int batchNumber)	If items in a batch were instantiated, and Keep Reference is true, attempt to destroy the instance of those items. If the item is an addressable, the instance will also be released. Numbers begin at 1.
DestroyBatchItems (LBSBatch batch)	If items in a batch were instantiated, and Keep Reference is true, attempt to destroy the instance of those items. If the item is an addressable, the instance will also be released.
GetBatchByID (int batchID)	Attempt to find a batch in the list using the BatchID.
GetBatchByNumber (int batchNumber)	Get the batch, if any, using the number in the list of batches. Numbers begin at 1 and go up to the number of batches in the list.
GetBatchByIndex (int batchIndex)	Get the batch, if any, using the zero-based index in the list of batches.
GetBatchByName (string batchName)	Attempt to find the batch in the list with the given Batch Name. This may affect Garbage Collection and performance. Where possible use GetBatchByIndex(..) or GetBatchByID(..)
Initialise()	Initialise the Batcher. Has no effect if already initialised.
InsertBatch (LBSBatch newBatch, int insertBeforeIndex)	Attempt to insert a batch into the list of batches.
PauseBatcher()	Attempt to pause the batcher.
RemoveBatch (LBSBatch batchToRemove)	Attempt to remove a batch from the list of batches.
RemoveBatchAt (int batchIndex)	Attempt to remove a batch at the zero-based index in the list of batches.
RunBatch (LBSBatch batch)	Attempt to run a single batch using the BatchID.
RunBatchByNumber (int batchNumber)	Attempt to run a single batch using the number in the list of batches. Numbers begin at 1.
RunBatchByID (int batchID)	Attempt to run a single batch using the BatchID.
RunBatchesAll()	Attempt to start all batches taking into consideration any delays.
UnloadBatchItems (LBSBatch batch)	Attempt to unload from memory any addressable item prefabs from a given batch. See also DestroyBatchItems(..).
UnloadBatchItems (int batchNumber)	Attempt to unload from memory any addressable item prefabs from a batch. See also DestroyBatchItems(..).
UnloadBatchItemsAll ()	Attempt to unload batch item addressable prefabs from all enabled batches.
UnPauseBatcher()	Attempt to unpause the batcher.

Batcher Methods – Static

These public methods can be accessed from LBSBatcher.

Method	Description
CreateBatcher()	Attempt to add a LBSBatcher gameobject and component to the active scene.

Downloader Methods - General

These public methods can be accessed from the instance of LBSDownloader.

Method	Description
AddContent (LBSContent newContent)	Attempt to add content to the end of the list of content.
CancelDownload()	Attempt to cancel the current download.
CheckForNewContent()	Check for new content to download,
ClearCacheAll()	Attempt to clear all the downloaded content cache.
DownloadContentAll()	Attempt to download all the new content available.
GetContentByID (int contentID)	Attempt to find content in the list using the ContentID.
GetContentByNumber (int contentNumber)	Attempt to find content given the number in the content list. Numbers begin at 1.
Initialise()	Attempt to initialise the component.
InsertContent (LBSContent newContent, int insertBeforeIndex)	Attempt to insert content into the list of content.

Dynamic Include Methods - General

These public methods can be accessed from the instance of LBDynanicInclude.

Method	Description
CancelPendingInitialise()	If waiting for Initialise() to start, cancel it now.
Initialise()	Attempt to initialise the component.

Dynamic Include Methods - Static

These public methods can be accessed from LBDynanicInclude.

Method	Description
CreateDynamicInclude()	Attempt to create a new DynamicInclude in the scene.

Include Helper Methods - General

These public methods can be accessed from the LBIncludeHelper at runtime.

Method	Description
Initialise()	Attempt to initialise this component. Has no effect if already initlalised.
SetKey (string newKey)	Attempt to set a new include helper key which can be used to match optional.
SetItemType (LBJumpManager.LBIncludeItemType newItemType)	Attempt to set the jump included item type.
SetResetPosition (Vector3 newResetPosition)	Update the reset position value which takes a localPosition value. this is a root game object, this will be the same as a world-space position.
SetResetRotation (Quaternion newResetRotation)	Update the reset rotation value which takes a localRotation value. If this is a root game object, this will be the same as a world-space rotation.

Jump Include Methods - General

These public methods can be accessed from the LBJumpInclude at runtime.

Method	Description
CancelPendingInitialise()	If waiting for Initialise() to start, cancel it now.
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 - Addressables

These public methods can be accessed from the LBJumpManager at runtime. To get the instance call LBJumpManager.GetManager() or LBJumpManager.GetOrCreateManager().

Method	Description
IsAddressableScene (AssetReference assetRef)	In the editor, is this asset a scene?
VerifyAddressable (AssetReference assetRef)	Verify that the addressable is valid. Currently this means that the runtime key is valid.

Jump Manager Methods - General

These public methods can be accessed from the instance of LBJumpManager. To get the instance call LBJumpManager.GetManager() or LBJumpManager.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 – FX General

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

Method	Description
GetCanvas()	Attempt to get the canvas used during jumps or transitions. See also GetOrCreateCanvas()
GetOrCreateCanvas()	Get or create a new child canvas used for fx, fade or progress during jumps
SetFXCanvasSortOrder (int newSortOrder)	Set the sort order in the scene of the FX canvas. Higher values appear on top.

Jump Manager Methods – Include and Scene Helpers

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

Method	Description
GetIncludeHelper (int includeHelperKeyId)	Attempt to get the first matching IncludeHelper with the give include helper key Id. This is not very performant, so use sparingly.
GetSceneHelper (LBJumpScene jumpScene, int includeHelperKeyId)	Attempt to get a scene helper attached to a JumpScene component. This is not very performant, so use sparingly.

Jump Manager Methods – Jump Includes

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

Method	Description
RegisterInclude (LBJumpInclude jumpInclude)	Attempt to register a Jump Include component with the Jump Manager.
UnRegisterInclude (LBJumpInclude jumpInclude)	Attempt to unregister a Jump Include component with the Jump Manager.

Jump Manager Methods - ProgressBar

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

Method	Description
SetProgressBar (LBSProgressBar newProgressBar)	Set a progressbar.

Jump Manager Methods - Scenes

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

Method	Description
JumpToDestination (LBJump 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 (LBJump jump, bool showErrors = true)	Verify if the standard scenes exist in the Unity Build Settings.

Jump Manager Methods – Static

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

Method	Description
GetActiveJump()	Get the jump (if any) that is in progress.
GetActiveJumpName()	Get the name of the jump (if any) that is in progress.

Method	Description
GetActiveSceneName()	Attempt to get the name of the active scene. Return "Unknown" if none active.
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.
GetProgessLoadDestination()	Attempt to get the progress (0-100) of loading the destination scene. If there is no jump in progress, this will return 0.
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). 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.
JumpNow (string jumpName)	Attempt to jump to a destination scene using the Jump Name. If you already have a reference to the Jump Manager, instead call JumpToDestination (jumpName).
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: LBJumpManager.RemoveListeners();

Jump Scene Methods - General

These public methods can be accessed from the LBJumpScene at runtime. To get the instance call LBJumpManager.GetManager().CurrentSourceJumpScene() or LBJumpManager.GetManager().CurrentDestJumpScene.

Method	Description
GetCameraType()	Get the type of camera type used during a jump in this scene
GetStandardCamera()	Get the current standard Unity camera.
Initialise()	This should be automatically called by the Jump Manager.
SetCameraType (LBJumpManager.LBJumpCameraType newCameraType)	SetCameraType (LBJumpManager.LBJumpCameraType newCameraType)
SetEventSystem (EventSystem newEventSystem)	Set the EventSystem used in this scene.
SetStandardCamera (Camera newCamera)	Use a standard Unity camera

Jump Scene Methods - Events

These public methods can be accessed from the LBJumpScene at runtime. To get the instance call LBJumpManager.GetManager().CurrentSourceJumpScene() or LBJumpManager.GetManager().CurrentDestJumpScene.

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 – Scene Helpers

These public methods can be accessed from the LBJumpScene at runtime. To get the instance call LBJumpManager.GetManager().CurrentSourceJumpScene() or LBJumpManager.GetManager().CurrentDestJumpScene.

Method	Description
GetFirstSceneHelper (int includeHelperKeyId)	First attached component (if any) that implements the ILBSSceneHelper interface and has the given KeyID

Jump Scene Methods – Transitions

These public methods can be accessed from the LBJumpScene at runtime. To get the instance call LBJumpManager.GetManager().CurrentSourceJumpScene() or LBJumpManager.GetManager().CurrentDestJumpScene.

Method	Description
SetFXFromCanvasPos (Vector3 newPosition)	Set a new FX from canvas position. Used when Render Mode is World Space.
SetFXToCanvasPos (Vector3 newPosition)	Set a new FX to canvas position. Used when Render Mode is World Space.
SetFXFromRenderMode (RenderMode newRenderMode)	Set the canvas render mode used when transitioning from this scene.
SetFXToRenderMode (RenderMode newRenderMode)	Set the canvas render mode when transitioning to this scene.

Jump Scene Methods – Static

These public methods can be accessed from the LBJumpScene at runtime.

Method	Description
GetDefaultFXFromColourCurve()	Get the default animation curve for transitioning between two colours in source scene.
GetDefaultFXFromCurve()	Get the default animation curve for transitioning from a scene.
GetDefaultFXToColourCurve()	Get the default animation curve for transitioning between two colours in destination scene.
GetDefaultFXToCurve()	Get the default animation curve for transitioning to a scene.
GetMirrorFXType (int fxTypeInt)	Attempt to get the mirrored LBSFXType as an integer. If there is no mirror, return the same type.
GetOrCreateJumpScene (int sceneHandle = 0)	Returns the current LBJumpScene instance for this scene. If one does not already exist, a new one is created. If the LBJumpScene is not initialised, it will be initialised. For multi-additive scenes, pass in the current scene handle e.g., gameObject.scene.handle.
IsWipeFX (int fxTypeInt)	Get a LBSFXType as an integer, determine if it is a Wipe FX.

Jump To Helper Methods - General

These public methods can be accessed from the instance in the scene at LBJumpToHelper at runtime.

Method	Description
InitiateJump (string jumpName)	Execute a jump with the jump name from the master JumpManager which may be in another scene.

Method	Description
InitiateJump (int jumpNumber)	Execute a jump with the jump number from the master JumpManager which may be in another scene. Numbers begin at 1.

Jump To Helper Methods - Static

These public methods can be accessed from the LBJumpToHelper at runtime.

Method	Description
CreateHelper()	Attempt to add a jump to helper gameobject and component to the active scene.
QuitGame()	Attempt to quit the game. This could be called by when a button is pressed in scene but could also be controlled by a menu that is located in the DontDestroyOnLoad scene.
StopMainCamera (bool isStopListener)	If there is a main camera, stop it and optionally the listener. NOTE: This has performance overhead so use sparingly.

Release History

Initial Release – 09 May 2025

Trade Marks