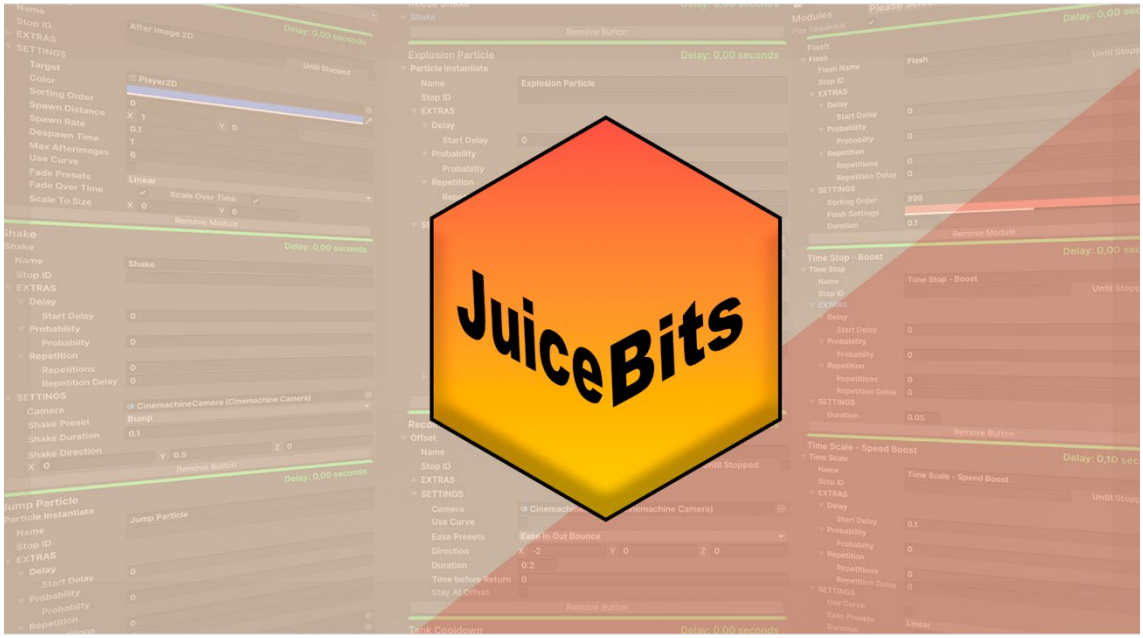
Documentation



03.10.2025

gamedev.kriegel@gmail.com

Table of contents

1 Welcome & Thanks

Welcome to my asset package about camera and particle game juice. I hope you enjoy the package.

2 Overview

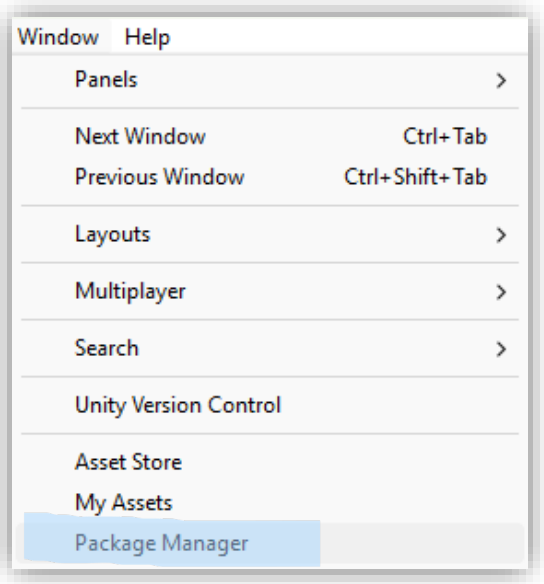
This documentation is a guide to explain the usage of the asset package to the user.

3 Dependencies

The package uses the **Cinemachine Camera** to create the game juice effects. Please be sure to have the Cinemachine installed.

4 Installation

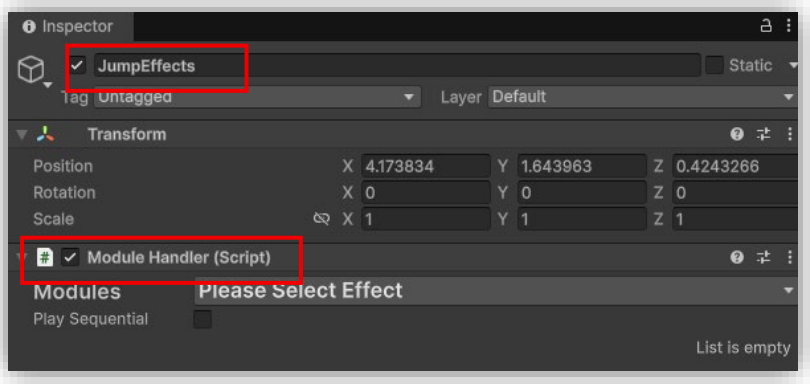
First of all, you must download the asset package from Unity's Asset Store. After this you start your Unity Project and go to **Window -> Package Manager -> My Assets**. There you can search for the **Asset Package Name** and **Install** and **Import** it to your project. If the **Cinemachine** does not get installed with the package you have to **manually add** it.



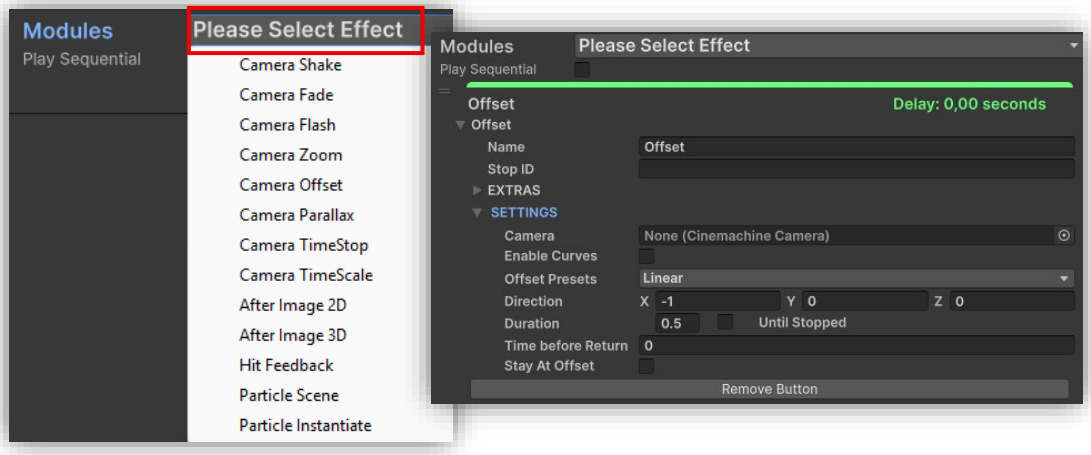
5 How to use

Brief overview of the asset package setup in your project.

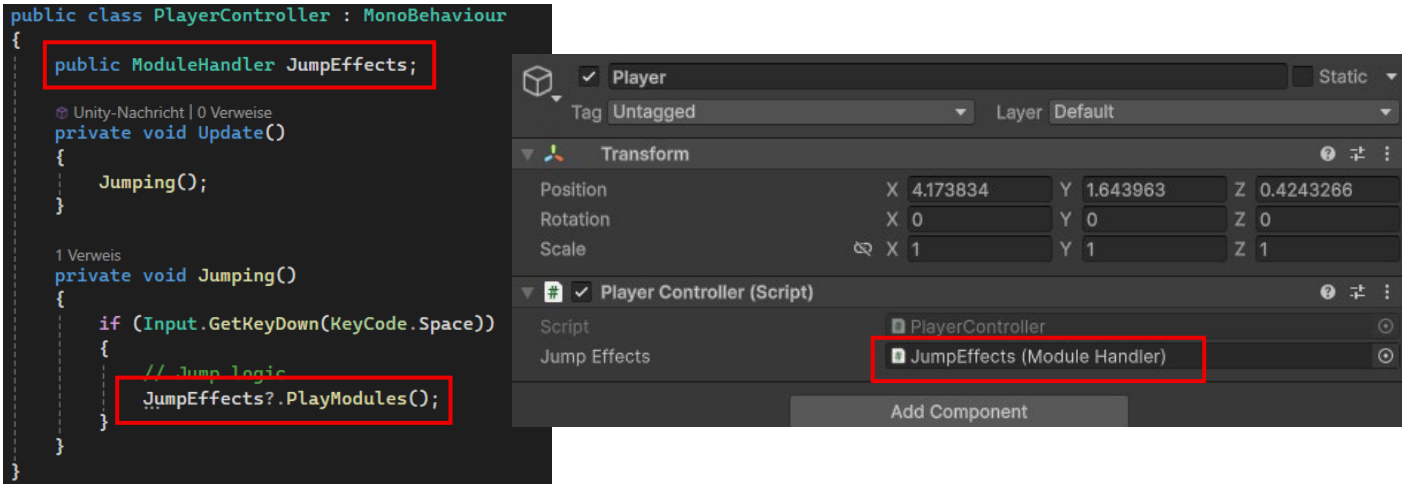
1. **Add** the **ModuleHandler** to an empty **GameObject**.



2. **Select** the desired **module** in the **dropdown menu**.

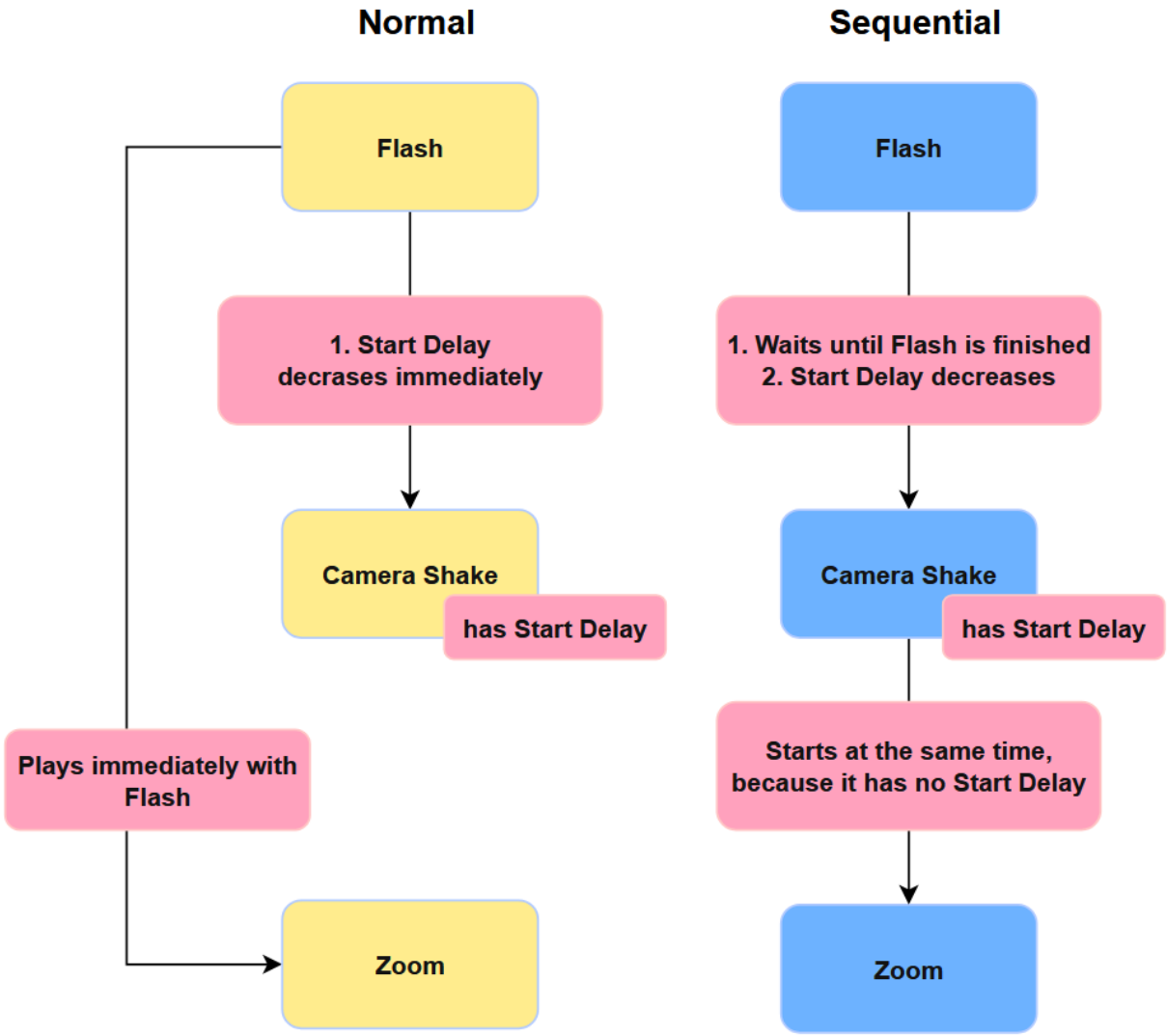


3. Add the **JuiceBits** namespace and **create** a **reference** in a **script** to the **ModuleHandler** and call **PlayModules**();.



6 What is Sequential Play

Sequential play waits for the module to finish before it starts the next one.



7 Modules

7.1 General Fields

Field	Usage
Name	Changes the name of the module
Stop ID	Stops the module with „StopModuleByID“
Until Stopped	Loops until stopped in script
Start Delay	Starts module with delay
Probability	Chance to trigger the module
Repetitions	Number of repetitions
Repetition Delay	Delay between repetitions

7.2 Camera Shake

Field	Usage
Camera	Target Cinemachine camera of the shake
Shake Preset	Presets to choose from
Duration	Duration of the shake
Direction	Changes direction of the shake

7.3 Camera Fade

Field	Usage
Use Curve	Enables the use of an animation curve
Ease Presets	Predefined curves that change a value over time
Fade Type	Choose between fade types
Sorting Order	Order of the objects drawn in the scene
Color	Color of the fade
Duration	Duration of the fade

7.4 Camera Flash

Field	Usage
Sorting Order	Order of the objects drawn in the scene
Color	Color of the flash
Duration	Duration of the flash

7.5 Camera Zoom

Field	Usage
Camera	Target Cinemachine Camera of the zoom
Use Curve	Enables the use of an animation curve
Ease Presets	Predefined curves that change a value over time
Target Zoom	Orthographic or perspective value to zoom at
Zoom Out Duration	Duration of the Zoom Out
Zoom in Duration	Duration of the Zoom In
Time Before Return	Time at the Target Zoom before the Zoom In starts

7.6 Offset

Field	Usage
Camera	Target Cinemachine Camera of the offset
Use Curve	Enables the use of an animation curve
Ease Presets	Predefined curves that change a value over time
Direction	Which direction the offset moves to
Duration	Duration of the offset
Time Before Return	Time at the offset direction before the return
Stay At Offset	The offset stays at the offset forever

7.7 Parallax

Field	Usage
Layer	The selected object/container that wants to be a parallax
Speed	The movement speed of the Parallax

7.8 Time Stop

Field	Usage
Duration	Duration of the time stop

7.9 Time Scale

Field	Usage
Use Curve	Enables the use of an animation curve
Ease Presets	Predefined curves that change a value over time
Instantly	Changes to the time scale instantly and back to original after duration
Duration	Duration of the time scale
Start Scale	Start scale to scale time from
End Scale	End scale to scale time to

7.10 Afterimage 2D & Afterimage 3D

Field	Usage
Target	Target of the afterimage creation
Color	Color of the afterimage
Spawn Distance	The offset to the target
Max Afterimages	Maximal numbers of afterimages in the scene at the same time
Spawn Rate	Time to create an afterimage
Despawn Time	Time before the afterimage disappears
Sorting Order	Order of the objects drawn in the scene
Fade Over Time	Enables fading of the afterimage over time
Scale Over Time	Enables scaling of the afterimage over time
Use Curve	Enables the use of an animation curve
Fade Presets	Predefined curves that change a value over time
Scale To Size	Decide the scaling values of the afterimage over time

7.11 Hit Feedback

Field	Usage
Target	Target of the hit feedback
Color	Color of the hit feedback
Duration	Duration of hit feedback

7.12 Particle Scene

Field	Usage
Target	Target of the particle systems
Particle List	List of the affected particle systems
Set As Parent	Sets the target as parent of the particle systems
Position Of Parent	Sets the position of the particle systems to the target position
Offset	The offset of the particle systems
Activate Particle	Enables the use of particle system activation
Activate Time	Time until the particle system gets activated
Activate At Index	Activate specific elements via element index of particle list
Deactivate Particle	Enables the use of particle system deactivation
Deactivate Time	Time until the particle system gets deactivated
Deactivate At Index	Deactivate specific elements via element index of particle list

7.13 Particle Instantiate

Field	Usage
Target	Target of the particle systems
Particle Prefab	Particle prefab to instantiate
Offset	The offset of the particle systems
Use Cooldown	Enables a time window between the next particle instantiation
Cooldown Time	The duration of the cooldown
Set As Parent	Sets the target as parent of the particle systems
Use Pooling	Enables the use of a pool
Default Capacity	The default size of the pool
Max Capacity	The maximum size of the pool, when needed

7.14 Particle System Randomizer

The Particle System Randomizer takes the **Min Value** and **Max Value** and **generates the User a random Value**. If the values did not get changed and „Use Randomizer“ is on, the standard values of the particle system are taken.

Field	Usage
Use Randomizer	Enables the use of min and max values
Min Emission	Min value of emisison
Max Emission	Max value of emission
Min Duration	Min value of duration
Max Duration	Max value of duration
Min Lifetime	Min value of lifetime
Max Lifetime	Max value of lifetime
Min Size	Min value of size
Max Size	Max value of size
Min Rotation	Min value of rotation
Max Rotation	Max value of rotation
Random Color	List of colors that get picked randomly every instantiation
Random Material	List of materials that get picked randomly every instantiation

8 Expanding the Asset-Package

Short overview on how to add your own modules to the system.

8.1 Module Script

First of all, you need a new **script** for a **new effect** that you want to implement this script must **inherit** from **ModuleBase** and have the whole logic.

Note: You can write your logic with the module still inheriting from MonoBehavior to quickly test it without first creating a custom UI. After it works use the ModuleBase parent class. Otherwise creating a custom UI and testing the module at the same time can be frustrating.

```
public class NewEffectModule : ModuleBase
{
    3 Verweise
    public override void Initialize(GameObject targetObject)
    {
        // The Awake of our Script
    }

    2 Verweise
    public override void Update()
    {
        // Update
    }

    3 Verweise
    public override void Play()
    {
        // Plays the Module logic at the end
    }
}
```

8.2 User Interface Script

In this script you create the Custom UI for your new effect.

```
public class NewEffectUI
{
    private NewEffectModule _module;
    private VisualTreeAsset _userInterface;

    0 Verweise
    public NewEffectUI(NewEffectModule module, VisualTreeAsset userInterface)
    {
        _module = module;
        _userInterface = userInterface;
    }

    0 Verweise
    public VisualElement CreateInterface(System.Action removeModule = null)
    {
        VisualElement root = _userInterface.CloneTree();

        // UI Logic

        return root;
    }
}
```

8.3 UI-Document

After that you create an **UI document (VisualTree)** and build the UI in the **UI Builder** from Unity or on your own.

When this is done you add the **VisualTree** and **dropdown entry** inside the **ModuleHandlerEditor** and **declare the location** of your **UI document**.

```
private VisualTreeAsset VisualTreeNewEffect;
```

```
    private Dictionary<string, Func<ModuleBase>> _dropdownEntries = new()
{
    {"Camera Shake", () => CreateInstance<ShakeModule>()},
```

```
private void OnEnable()
{
    // Loads all UXML to work both in Edit and Play mode - UXMLs need to be in the Resources Folder
    VisualTreeNewEffect = Resources.Load<VisualTreeAsset>("UXML/Handler/ModuleNewEffectVisualTree");
```

After that you need to **add your module** to the **CreateInspector()** Method like below.

```
if (module is ShakeModule shakeModule)
    element.Add(new ShakeUI(shakeModule, VisualTreeShake).CreateInterface(RemoveModule));
```