NSR - Screen Recorder Documentation

Silver Tau

Plugins

2025

Content

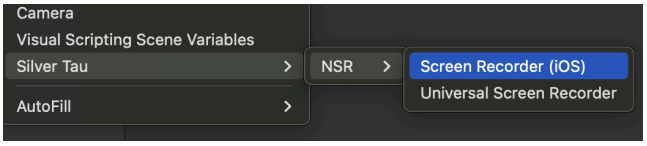
Overview

Features

Feature	Description
ScreenRecorder	The main component that creates a connection between the Unity engine and the screen recording framework. (Only iOS!)
UniversalVideoRecorder	The main component that creates a connection between the Unity Engine and the screen recording plugins (Android & iOS & macOS & Windows).
MicrophoneAudioRecorder	A component that creates a connection between the Unity Engine and audio recording plugins and provides the ability to create audio files (Android & iOS & macOS & Windows).
GraphicProvider	The main system that manages subsystems, parameters, and the way images are created and saved.
GraphicSettings	A script object that sets the general settings of the system. GraphicSubsystem.
GraphicSubsystem	An abstract class that is the basis for creating a subsystem that provides the original image.

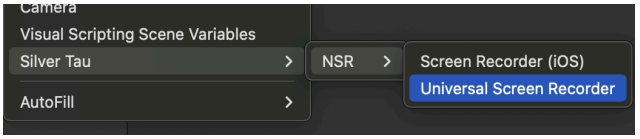
Adding a NSR (Native Screen Recorder)

To add a basic module to a scene, you need to do the following:
1. In the scene hierarchy, add a component from the quick menu (or manually).
2. In the Component inspector, add the basic elements.



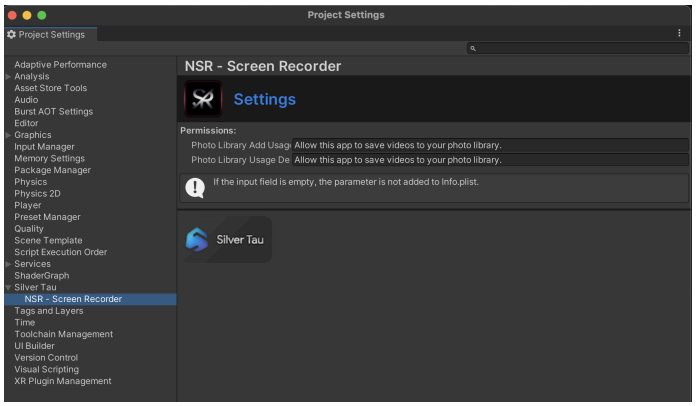
Adding a NSR (Universal Screen Recorder)

To add a basic module to a scene, you need to do the following:
1. In the scene hierarchy, add a component from the quick menu (or manually).
2. In the Component inspector, add the basic elements.



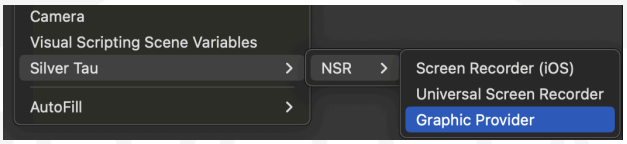
NSR (Screen Recorder & Universal Screen Recorder) settings

The NSR plugin also has permission settings for iOS Info.plist. To use them, go to Player Settings -> Silver Tau -> NSR - Screen Recorder.



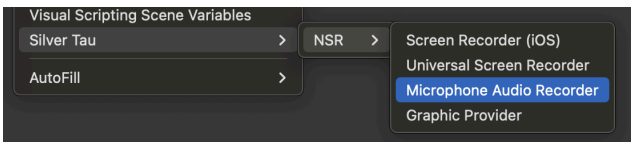
Adding a NSR Graphic Provider

To add a basic module to a scene, you need to do the following:
1. In the scene hierarchy, add a component from the quick menu (or manually).
2. In the Component inspector, add the basic elements.



Adding a Microphone Audio Recorder

To add a basic module to a scene, you need to do the following:
1. In the scene hierarchy, add a component from the quick menu (or manually).
2. In the Component inspector, add the basic elements.

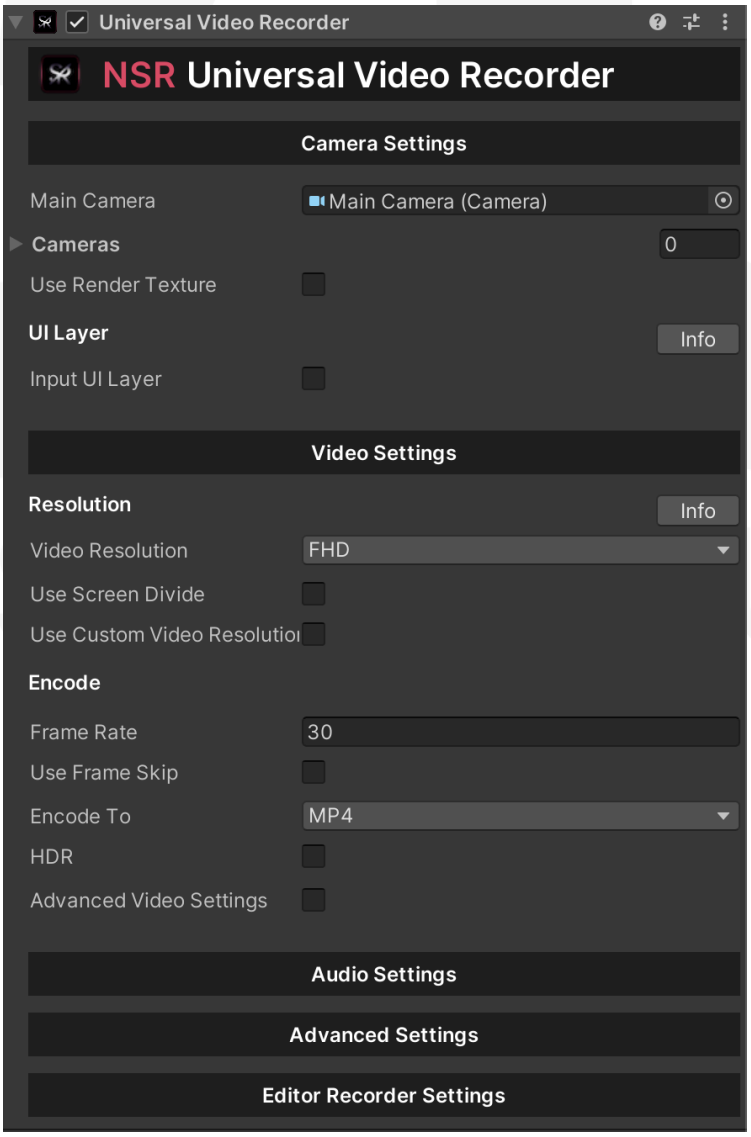


Universal Video Recorder

This is a cross-platform script that provides the ability to record video (Android & iOS & macOS & Windows).
The main component that creates the connection between the Unity engine and the NSR plugin.

Features:

- **High-speed:** Designed and extensively optimized for superior performance.
- **Capture Everything:** Capture any visual content that can be converted to texture, whether it's a game interface, user interface, camera image, or texture.
- **Custom resolution:** Capture video in resolutions up to Full HD (1920x1080) or even higher if your device supports it.
- **Augmented Reality Support:** The package provides full compatibility with ARFoundation, ARCore, ARKit and Vuforia.
- **Simultaneous recording:** The package provides thread security by allowing recording in worker threads to further improve performance.
- **Easy integration**: The API is purposefully designed to minimize unnecessary additions or additional burden on your project.



Actions

Property	Description
onStartCapture	Action that is triggered when you start recording a video.
onPauseCapture	Action that is triggered when video recording is paused.
onResumeCapture	Action that is triggered when video recording is resumed after pausing.
onStopCapture	Action that is triggered when you stop recording a video.
onErrorCapture	Action that is triggered when a video recording error occurs.
onCompleteCapture	Action that is called after a video is successfully created.
onOutputVideoPathUpdated	Action that is called after a video is successfully created and returns the updated output path of the video file.
onOutputSeparateAudioPathUpdated	Action that is called after an audio is successfully created and returns the updated output path of the audio file.
onFrameRender	An action that is triggered every frame when recording a video. This frame is provided before it enters the video record, so it can be modified.

Task

Property	Description
GetVideoOutputPath	A task that returns the original path of a video file asynchronously.

Properties

Property	Description
mainCamera	The main rendering camera.
cameras	List of rendering cameras. It can be used to dynamically switch recording cameras.
useRenderTexture	A parameter that indicates that the targeted Render Texture will be used for video recording.
targetRenderTexture	The target RenderTexture.
CustomOutputVideoFilePath	Custom path to the video output file. Does not include the file

	name. If the path is empty, Application.persistentDataPath is used.
CustomOutputVideoFileName	A custom name for the video output. The file format will be added automatically. If the name is empty, automatic name generation is used. **Example:** video_yyyy_MM_dd_HH_mm_ss_fff.mp4
CustomOutputSeparateAudio FilePath	Custom path to the audio output file. Does not include the file name. If the path is empty, Application.persistentDataPath is used.
CustomOutputSeparateAudio FileName	A custom name for the audio output. The file format will be added automatically. If the name is empty, automatic name generation is used. **Example:** audio_yyyy_MM_dd_HH_mm_ss_fff.wav
layerMasks	Layers that are displayed during recording.
videoResolution	Target video resolution. The video resolution determines the amount of detail in your video, or how realistic and clear it looks. **Warning**: Be careful with high values, as they may not be supported on your device or may cause the app to slow down during recording.
useScreenDivide	Use a divider to further resize the recording on the screen.
screenDivide	Divider for the size of the screen recording.
useCustomVideoResolution	The custom video resolution option allows you to set any value.
customVideoResolution	Custom video resolution. Just be careful with high values.
recordMicrophone	Recording sound from a microphone? A parameter that indicates whether the sound from the microphone will be recorded.
encodeTo	Video output encoding format.
HDR	HDR (high dynamic range) is used for a wide range of colors. Use this option when using HDR colors for materials or post-processing (e.g., Bloom effect).
videoRecorderManager	Video recorder manager (preview).
useRealtimeClock	Do we use a runtime clock? A parameter that indicates whether the timer will be counted when recording video.
textTimer	A text component that displays the recording time.
VideoOutputPath	Video output path.

IsRecording	Is it recording? A setting that indicates whether video is being recorded.
GetRecordStatus	A parameter that allows you to find out the status of a record.
frameRate	Set the custom frame rate. Default - 30 fps.
frameSkip	Control number of successive camera frames to skip while recording. Default value = 0.
autoPauseResumeRecorder	Automatically pause/resume video recording during program focus/pause
sampleRate	Sample rate is the number of samples per second that are taken of a waveform to create a discete digital signal. Default value = 48000.
channelCount	The channelCount property is an enumerated value that describes how channels should be mapped between the inputs and outputs of the node. Default value = 2.
audioBitRate	Bitrate is a term used to describe the amount of data transmitted in audio. The higher the bitrate, the better the sound quality. Default value = 96_000.
ConfigInputUILayer	An option that indicates that the user interface layer will be recorded. Option for real-time interaction.
inputUILayer	An option that indicates that the UI layer is about to be recorded.
videoBitRate	Bitrate refers to how many bits can be transferred or processed within a certain amount of time. Default value = 10000000.
recordSeparateAudioFile	An option that allows you to save a separate audio file of the recording.
separateAudioFileFormat	The format of the output audio file
frameDescriptorColorFormat	Defines the color format of the RenderTexture used for custom frame processing. For example, ARGB32 provides 8-bit per channel (32-bit total) with alpha.
useFrameDescriptorSrgb	When true, enables sRGB color encoding for the RenderTexture, which applies gamma correction. Set this to false if you're working in linear color space and need precise control over pixel values (e.g. for post-processing).
frameDescriptorMsaaSamples	Sets the multi-sample anti-aliasing (MSAA) level. A value of 1 disables MSAA. You can increase it to reduce edge aliasing (e.g., 2, 4, 8), but note that higher values increase GPU cost and memory usage.

Functions

Property	Description
StartVideoRecorder	A method that starts recording the screen. <param name="videoFilePath">Custom path.</param> <param name="videoFileName">Custom file name.</param> <param name="audioFilePath">Custom path.</param> <param name="audioFileName">Custom file name.</param>
StopVideoRecorder	A method that stops recording the screen.
PauseVideoRecorder	A method that pauses screen recording.
ResumeVideoRecorder	A method that resumes screen recording.
CreatePreviewImage	A method that converts RenderTexture and returns Texture2D. <param name="width">Target width.</param> <param name="height">Target height.</param> <param name="callback">Callback action.</param>
FindActiveTargetCamera	An additional method that searches for the active main camera.
ClosestInteger	A method that helps you express the size of a multiple of a number. <param name="a"> Value input.</param> <param name="b"> A multiple of a number.</param>
ScreenDivide	A method that helps to express the target screen size for video recording.
SaveVideo	A method that performs the function of storing video in the device's gallery. ***Use your own methods to save the video. The final recording path corresponds to the VideoOutputPath parameter.*
Preview	A method that performs the preview function.
ShareVideo	A method that performs the share function. ***Use your own methods to share the video. The final recording path corresponds to the VideoOutputPath parameter.*

Record video without UI layer

Standard rendering type (Standard).
If you want to record the screen content along with the UI content, the Canvas interface needs to display the camera content.

To do this, do the following

1. Enable the InputUILayer option.
2. In the Canvas settings, set the rendering mode to Screen Space - Camera or World Space. This will allow you to add an interface to the content that your main camera is rendering on stage.

If you want to record the screen content without the UI content, remember that the Canvas interface needs to display the content on top of the camera.

To do this, do the following

1. turn off the InputUILayer option.
2. in the Canvas settings in the viewer, select View Mode - Screen Space - Overlay.

If you plan to change the state of the UI layer in real time, use the UniversalVideoRecorder.ConfigInputUILayer parameter and change the Render Mode of the target canvas. For an example, see the NSR - Screen Recorder_Record UI.

URP (Universal Render Pipeline)
If you want to record screen content without UI content you need to:

1. UI Canvas should display content on top of the camera. To do this, set the mod - Screen Space - Overlay in the canvas settings in the renderer.
2. Add a camera to the main camera stack that will render only the UI layer.
3. Set the recording layers you need in the Universal Video Recorder script. For example, remove the UI layer.

The entire screen recording is ready for you to record the screen of the device without the content of UI components.

Save video and audio files

Saving the video and audio file is fully automated and has options for custom changes. Below are some examples of how a file is saved, its path, name, and how you can customize it.

Note	You can specify the path and name for the output files when you start recording video using the **StartVideoRecorder(var path, var name, ...)** function.

Automatic saving

Automatic file saving is designed to avoid crashes when the file path and name are empty and to simplify the saving process.
The default path for automatic saving is **Application.persistentdatapath**.
The name of the output file for automatic saving is formed as follows:
video + time + format.
Example: video_yyyy_MM_dd_HH_mm_ss_fff.mp4.

Custom saving

Custom file saving allows you to set any path or name for the output file.

To change the output path of the video file, use the **“CustomOutputVideoFilePath”** option.

Example of use:

```
public class CustomSaving : MonoBehaviour
{
  public void Run()
  {
    UniversalVideoRecorder.CustomOutputVideoFilePath = Path.Combine(Application.
persistentDataPath, "New folder");
    UniversalVideoRecorder.StartVideoRecorder();
  }
}
```

Note	If the output path is empty, an automatic path generation will be applied. (**Application.persistentdatapath**).

To change the name of the output video file, use the **“CustomOutputVideoFileName”** option.

Example of use:

```
public class CustomSaving : MonoBehaviour
{
  public void Run()
  {
    UniversalVideoRecorder.CustomOutputVideoFileName = "new name";
    UniversalVideoRecorder.StartVideoRecorder();
  }
}
```

Note	If the output file name remains the same, automatic name generation will be applied. (**video + time + format**)

Dynamic change of cameras

In order to dynamically change cameras during video recording, you need to add the cameras you need to the “Cameras” list in Universal Video Recorder before starting video recording. After creating the list of cameras, the last camera that was turned on will be recorded. To switch cameras, you can simply turn the camera you need on or off.

Note	An example of using dynamic camera changes can be found in the **NSR scene - Screen Recorder_Dynamic Camera Change**s.

Device permissions

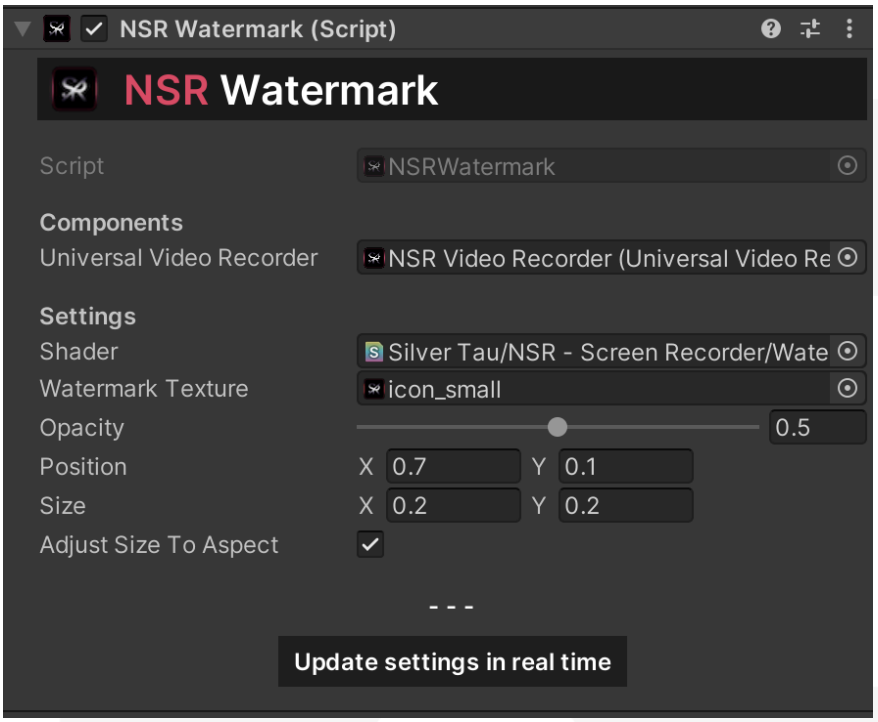
Starting with the 1.6.0 version of the plugin, the function of limiting the camera and microphone usage permissions has been added. Now you can use the plugin without requiring permission to use the microphone and camera, for example, if you only want to record video.

In order to remove the forced permissions for your application, you need to add the symbols you need to **Scripting Define Symbols** in **Player Settings -> Other Settings**.

Scripting Define Symbols:
NSR_MICROPHONE_DISABLE - allows you to completely disable the device's microphone (for the plugin). **NSR_CAMERA_DISABLE** - allows you to completely disable the device's camera (for the plugin).

Watermark

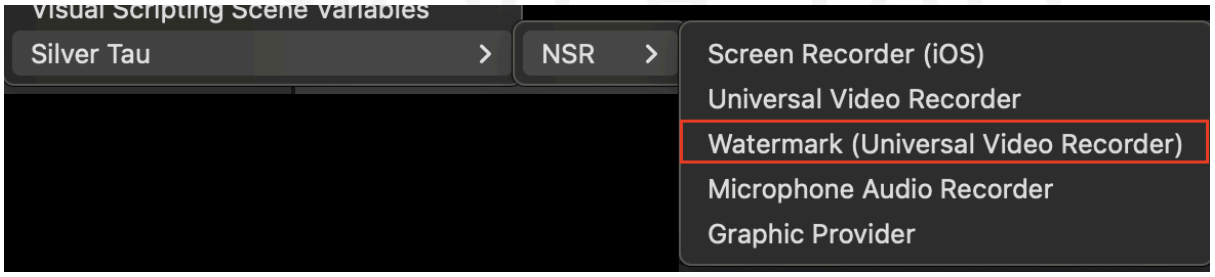
Starting with the 1.7.2 version of the plugin, the ability to process frames before sending them to the recorder has been added. The NSR - Watermark component is responsible for applying a watermark to each frame of the video recorded with **UniversalVideoRecorder**. This component allows you to customize the appearance of the watermark using a **shader**, **texture**, **size**, **position**, and **transparency**. It also supports automatic scaling of the watermark to fit the aspect ratio of the texture.



Adding a NSR - Watermark

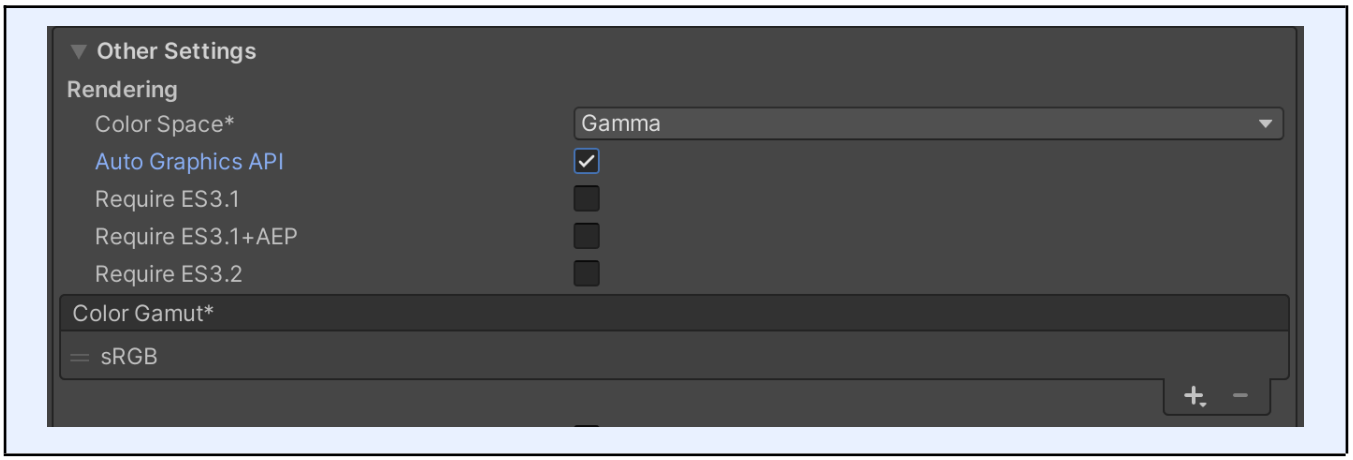
To add a basic module to a scene, you need to do the following:

1. In the scene hierarchy, add a component from the quick menu (or manually).
2. In the Component inspector, add the basic elements.



Note:

If you encounter black frames when recording video on **Android**, please check your Graphics API. We recommend setting the **Auto Graphics API** option.



How It Works

1. **Initialization**:
   - When the component is enabled (`OnEnable`), it creates an instance of `WatermarkRenderer` with the provided shader and parameters.
   - It subscribes to the `onFrameRender` event of the `UniversalVideoRecorder`.
2. **Watermark Rendering**:
   - On every video frame, `OnFrameRender` is triggered, applying the watermark texture on top of the video frame.
   - If `adjustSizeToAspect` is enabled, the watermark size is automatically adjusted to preserve the texture's aspect ratio (only done on the first frame).
3. **Live Updates**:
   - You can update the watermark's appearance at runtime using the following methods:
     - `ChangePosition(Vector2)` – updates position (normalized)
     - `ChangeSize(Vector2)` – updates size (normalized)
     - `ChangeOpacity(float)` – updates transparency (0–1)
     - `ChangeWatermarkTexture(Texture)` – changes the texture
     - `ChangeShader(Shader)` – applies a new shader
     - `UpdateSettingsRealtime()` – re-applies all current settings
4. **Cleanup**:
   - When the component is disabled (`OnDisable`), it unsubscribes from the render event and disposes of the watermark renderer.

Custom Frame

The **NSR - Watermark** component demonstrates how to inject custom logic into the video rendering pipeline using the `UniversalVideoRecorder.onFrameRender` event. This opens the door to building your **own custom frame processors**— for adding filters, effects, overlays, analysis, or any real-time modification to the captured video frames.

How to Create a Custom Frame Processor

To implement your own custom frame logic:

1. **Subscribe** to the `onFrameRender` event of the `UniversalVideoRecorder`.
2. **Process the frame texture** however you need (e.g., apply a shader, copy to another texture, analyze pixel data, etc.).
3. **Optionally render** your result back to the frame or save it elsewhere.

Example: Creating a Grayscale Frame Filter

Below is a simple example of a component that applies a custom grayscale shader to each video frame:

```
using UnityEngine;
using SilverTau.NSR.Recorders.Video;

public class CustomGrayscaleFrameProcessor : MonoBehaviour
{
    [SerializeField] private UniversalVideoRecorder recorder;
    [SerializeField] private Shader grayscaleShader;

    private Material _material;

    private void OnEnable()
    {
        if (recorder == null || grayscaleShader == null) return;

        _material = new Material(grayscaleShader);
        recorder.onFrameRender += OnFrameRender;
    }

    private void OnDisable()
    {
        if (recorder == null) return;

        recorder.onFrameRender -= OnFrameRender;
        Destroy(_material);
    }

    private void OnFrameRender(Texture frame)
```

```
    {
        RenderTexture temp = RenderTexture.GetTemporary(frame.width,
frame.height);
        Graphics.Blit(frame, temp, _material); // Apply grayscale shader
        Graphics.Blit(temp, frame);            // Overwrite original frame
        RenderTexture.ReleaseTemporary(temp);
    }
}
```

Use Cases for Custom Frame Logic

- Apply visual effects (grayscale, blur, sepia, distortion)
- Add dynamic overlays (text, UI, data feeds)
- Perform real-time computer vision or image analysis
- Stream frames to an external server

Performance Tip

Make sure your custom frame logic is efficient. It runs **every frame** and can easily bottleneck performance or cause dropped frames if:

- You allocate too many temporary textures
- Your shader is complex
- You perform CPU-GPU readbacks

Use `RenderTexture.GetTemporary()` **and** `Graphics.Blit()` wisely, and dispose of everything when done.

Screen Recorder (native iOS)

The main component that creates the connection between the Unity engine and the NSR plugin. (Only iOS)

Actions

Property	Description
recorderStart	An action that signals the start of screen recording.
recorderStop	An action that signals when the screen recording stops.
recorderShare	An action that signals that you have shared a recording.
recorderError	An action that signals an error when recording a screen.
recorderShareError	An action that signals an error when you share a recording.
onRecorderStatus	An action that is called when the screen recording status changes.

Properties

Property	Description
useDelayInitialization	If the parameter is enabled, initialization will occur only when you initialize the framework manually.
useMicrophone	This option allows you to turn the microphone on/off before recording a video.
usePopoverPresentation	This option allows you to enable/disable the presentation pop-up window after recording a video. This window also has

	its own built-in video editor.
useSaveVideoToPhotosAfterRecord	This option allows you to automatically save videos to the device's gallery after recording. If the usePopoverPresentation option is enabled, auto-saving will not work! Because the save window is called, which has the finish editing function - save or share the video file.
videoOutputPath	A parameter that returns the path of the output video file.
NSRInit	A parameter that indicates whether the package is initialized.
IsAvailable	A parameter that checks whether it is possible to record video on this device.
IsRecording	A parameter that checks whether the video recording is in the recording state.
GetVersion	A parameter that returns the package version value.
CurrentRecorderStatus	A parameter indicating the current state of screen recording.

Functions

Property	Description
Initialize	A method that initializes the framework.
Dispose	A method that disposes the framework.
UpdateSettings	A method that helps you update the settings for recording live video. <param name="microphone">Microphone status.</param> <param name="popoverPresentation">State of the popover presentation.</param> <param name="saveVideoToPhotosAfterRecord">Save video to a photo after recording?</param>
StartScreenRecorder	A method that starts recording the screen.
StopScreenRecorder	A method that stops recording the screen.
Share	A method that allows you to share. <param name="path"> Path to video. If the value is null, the value of the last recorded video is taken.</param>
SaveVideoToPhotosAlbum	A method that allows you to save a video file to Photos Album. <param name="path"> Path to video. If the value is null, the value of the last recorded video is taken.</param>

Microphone Audio Recorder

A component that creates a connection between the Unity Engine and audio recording plugins and provides the ability to create audio files (Android & iOS & macOS & Windows).

Actions

Property	Description
onCompleteCapture	An event indicating that the recording is complete.
onErrorCapture	An event that indicates that an error has occurred.
sampleBufferDelegate	An event that allows you to get the current sample buffer.

Properties

Property	Description
audioFormats	The format of the output audio file.
HeaderSize	The header size for the audio file. Be careful when changing it.

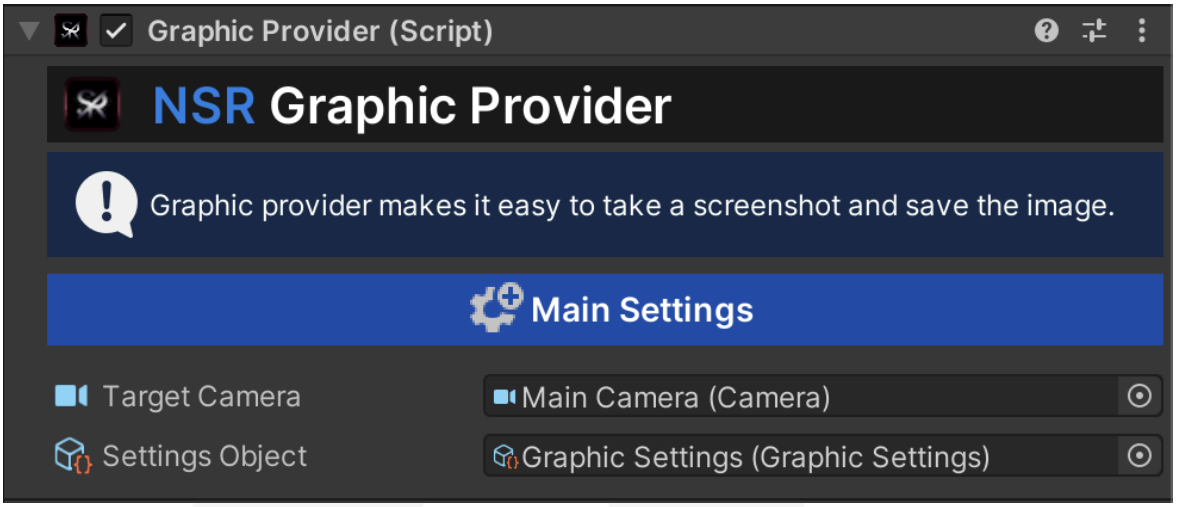
	The default value is 44.
setAutoFrequency	Automatic frequency detection.
frequency	Frequency for the audio file. Be careful when changing it. The default value is 48000.
setAutoChannels	Automatic detection of the number of channels.
channels	The number of channels for the audio file. Be careful when changing it. The default value is 2.
computeRMS	Enable/disable RMS value calculation.
computeDB	Enable/disable Decibel value calculation.
bufferWindowLength	The size of the buffer window for calculating RMS and Decibels.
CurrentRMS	Current RMS value.
CurrentDB	Current Decibel value.
CurrentAudioInputComponent	The current recording settings of the AudioInputComponent.
GetAudioSource	Audio Source to store Microphone Input, An AudioSource Component is required by default

Functions

Property	Description
GetMicrophoneDevices	Current microphone array.
StartRecording	A function that starts recording an audio file. <param name="micDeviceName">Target microphone name for recording.</param>
StopRecording	The function of ending the audio recording. <param name="filePath">The output path of the audio file.</param> <param name="fileName">The name of the audio file.</param>

Graphic Provider

The main system that manages subsystems, parameters, and the way images are created and saved. The root system is the main and unifying one for all subsystems.



Actions

Property	Description
OnCreateImage	An action that is called after a screenshot is successfully created.

Properties

Property	Description
mainCamera	The main rendering camera.
graphicSettings	This is a parameter that contains a link to a script object that sets the general settings for the system.
GraphicSubsystems	List of subsystems that perform the image rendering process.

Functions

Property	Description
CreateImage	A function that creates an image using a targeted group of subsystems that provide an image.

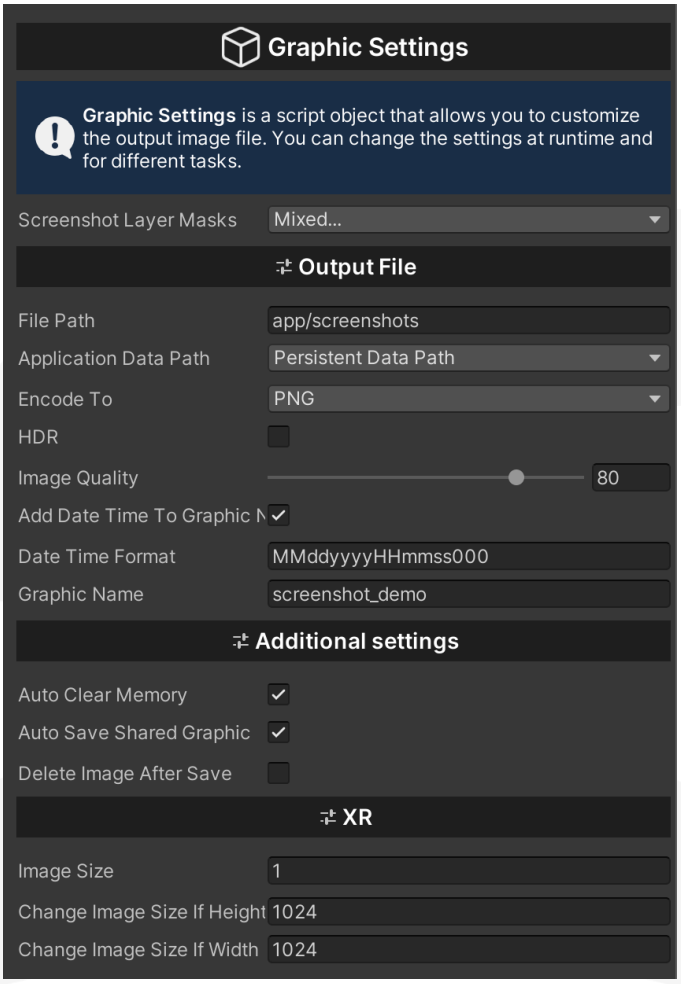
Examples of use:

Get an output path for shared graphics:

```
public class CustomScript : MonoBehaviour
{
  private void GraphicProcess(GraphicProvider graphicProvider)
  {
    if(graphicProvider.GraphicSubsystems == null) return;
    var graphicSubsystem = graphicProvider.GraphicSubsystems.First();

    if (graphicSubsystem == null)
    {
        Debug.Log("Graphic subsystem is missing.");
        return;
    }

    var sharedGraphic = graphicSubsystem.sharedGraphic;

    if (sharedGraphic == null)
    {
        Debug.Log("Shared Graphic is missing.");
        return;
    }

    graphicSubsystem.OnSharedGraphicSaved = path =>
    {
        // Output file path => path
        WriteGraphicInfo(sharedGraphic);
    };
  }

  private void WriteGraphicInfo(SharedGraphic sharedGraphic)
  {
    Debug.Log(string.Format("Shared Graphic: {0}", sharedGraphic.id));
    Debug.Log(string.Format("Name: {0}", sharedGraphic.name));
    Debug.Log(string.Format("Output path: {0}", sharedGraphic.outputPath));
  }
}
```

Graphic Settings

It is a script object that sets the general settings of the system. Settings can be changed both at runtime and in the Unity Editor. Also, the settings script can be changed at any time you need it.



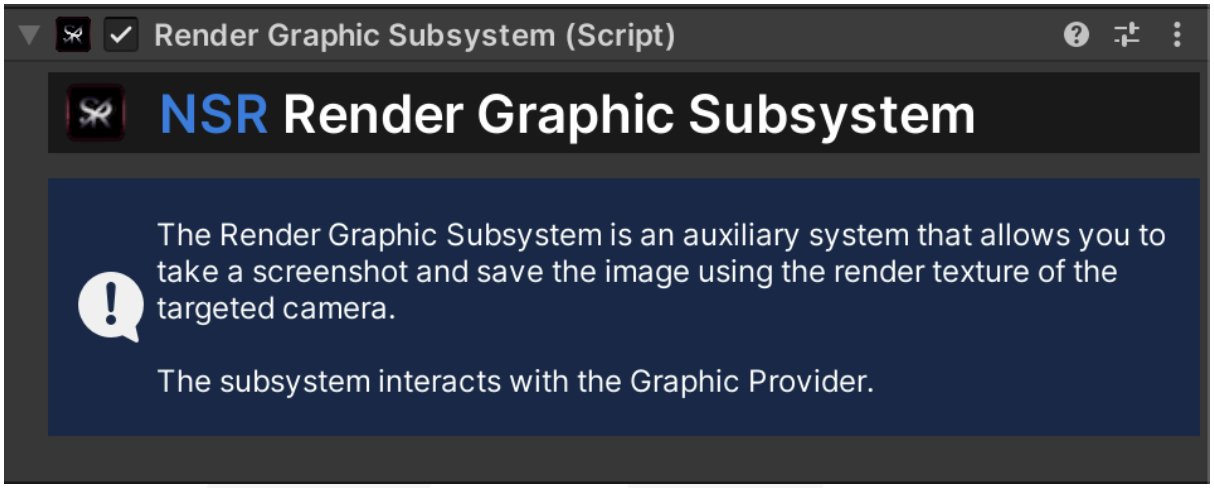
Properties

Property	Description
screenshotLayerMasks	Layers that will be displayed when you take a screenshot.
filePath	The part of the original path that will be used to save the image.
applicationDataPath	The main path where the images will be stored.
encodeTo	Image encoding format.
HDR	HDR (high dynamic range) is used for a wide range of colors. Use this option when using HDR colors for materials or post-processing (e.g., Bloom effect).

imageQuality	Graphic image quality. Only for .jpg format.
addDateTimeToGraphicName	A parameter that allows you to use your own time format for a screenshot.
dateTimeFormat	Custom time format for screenshots.
graphicName	A parameter that gives a custom name to the image.
autoClearMemory	A parameter that allows you to automatically clear the memory after the subsystem creates an image.
autoSaveSharedGraphic	A parameter that allows you to automatically save images after they are created by the subsystem.
deleteImageAfterSave	A parameter that allows you to automatically delete images after the subsystem creates an image.
imageSize	Sets the divider for the output image from the XR camera. 1 is the default value, which represents the original image size.
changeImageSizeIfWidthMore	Check the maximum output size of the image by width. If the image is larger than this check, the output image divider "imageSize" will be used.
changeImageSizeIfHeightMore	Parameter to check the maximum output image size in height. If the image is larger than this check, the output image divider "imageSize" will be used.

Graphic Subsystem

An abstract class that is the basis for creating a subsystem that provides the original image. With this class, you can create your own subsystems or modify existing subsystems to meet your needs.



Properties

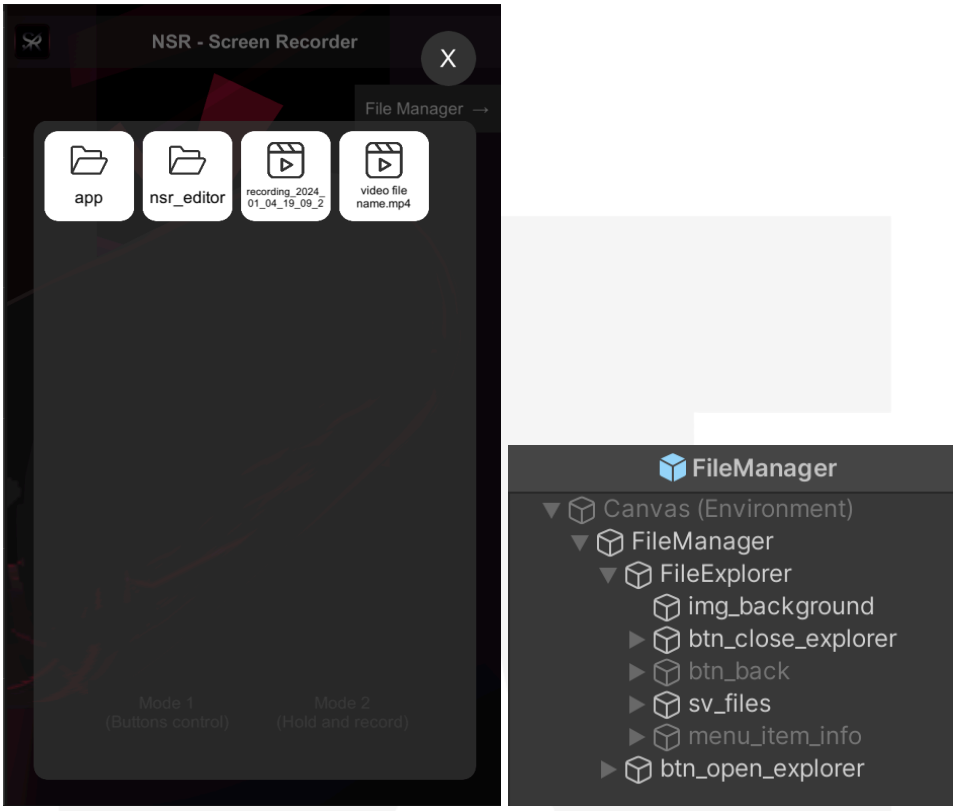
Property	Description
Id	Identifier of the graphics subsystem.
sharedGraphic	Shared Graphic which is formed from the subsystem.
RenderCamera	Targeted rendering camera for the subsystem.
GraphicSettings	This is a parameter that contains a link to a script object that sets the general settings for the system.
OnSharedGraphicSaved	An action that signals that the shared graphics file has been saved and returns the path of the saved file.

Functions

Property	Description
CreateImage	An abstract function that creates an image from the targeted subsystem.
SaveSharedGraphic	An abstract function that stores images from a targeted shared graphic.

File Manager Utility

An abstract class that is the basis for creating a subsystem that provides the original image. With this class, you can create your own subsystems or modify existing subsystems to meet your needs.



A file manager is a prefab that provides a user interface for managing files and folders. The most common operations performed on files or groups of files include opening (e.g., viewing, playing), deleting, and searching for files. Folders and files can be displayed in a hierarchical tree based on their directory structure.

Properties

Property	Description
storageType	The target storage type determines the location from which files will be parsed.
customStorageType	Custom path for file analysis and storage type. The parameter will be effective if the Storage type is set to Custom.
fileExplorer	File Explorer is a script that performs file recognition and search functions for the selected storage type.

Share Utility

Your app can share any files and folders. Share utility is a simple way to share content from one app to another, even from different developers.

Functions

Property	Description
ShareItem	A function that allows you to share an item. <param name="path"> Path to the item (file, folder, any path).</param>

Examples of use:

Share any file:

```
// A function that allows you to share a file & folder & image & video and e.t.c.
public void ShareFile(string path)
{
  Share.ShareItem(path);
}
```

Share Texture2D:

```
public void ShareTexture(Texture2D texture)
{
  if(texture == null) return;
  texture.Share(TextureEncodeTo.PNG);
  // or
  if (ShareExtension.TryShareItem(texture, TextureEncodeTo.PNG))
  {
      Debug.Log("Done!");
  }
}
```

Share Bytes:

```
public void ShareBytes(byte[] data)
{
  if(data == null) return;
  data.Share(".<file format>");
  // or
  if (ShareExtension.TryShareItem(data, ".<file format>"))
  {
      Debug.Log("Done!");
  }
}
```

Gallery Utility

Your app can save any videos and pictures to your device's gallery. The Gallery utility is a simple and convenient way to save content to your target device.

Actions

Property	Description
onSaveImageToGallery	An action that is called when the process of saving an image to the gallery is complete.
onSaveVideoToGallery	An action that is called when the process of saving a video to the gallery is complete.

Functions

Property	Description
SaveVideoToGallery	A method that allows you to save a video file to Gallery (Photos Album). <param name="path"> Path to video.</param> <param name="androidFolderPath"> Path to folder in android.</param>
SaveImageToGallery	A method that allows you to save an image file to Gallery (Photos Album). <param name="path"> Path to image.</param> <param name="androidFolderPath"> Path to folder in android</param>

Save the video and image file:

```
// A method that allows you to save a video file to Gallery (Photos Album).
public void SaveVideoFile(string path, string androidFolderPath)
{
  Gallery.SaveVideoToGallery(path, androidFolderPath);
}

// A method that allows you to save an image file to Gallery (Photos Album).
public void SaveImageFile(string path, string androidFolderPath)
{
  Gallery.SaveImageToGallery(path, androidFolderPath);
}
```

Save to gallery (Texture2D):

```
public void SaveToGalleryTexture(Texture2D texture)
{
   if(texture == null) return;
   texture.SaveImageToGallery(TextureEncodeTo.PNG);
   // or
   if (GalleryExtension.TrySaveImageToGallery(texture, TextureEncodeTo.PNG))
   {
       Debug.Log("Done!");
   }
}
```

Save to gallery (Bytes):

```
public void SaveToGalleryBytes(byte[] data)
{
   if(data == null) return;
   data.SaveVideoToGallery(".mp4");
   // or
   if (GalleryExtension.TrySaveVideoToGallery(data, ".mp4"))
   {
       Debug.Log("Done!");
   }
}
```

FAQ

Find answers and solutions to your questions. You can find answers to most questions or problems in the online documentation. For more details, please refer to the **FAQ**.