Readme - Screenshot Helper Free 1.3.8

Screenshot Helper is a powerful screenshots toolkit for almost all platforms, all Unity render pipelines, and supports capture images in several ways including fullscreen capture, camera capture, cutout area, specific size, scale, and with/without UI. Fit for any project to create accurate screenshots in the runtime app or Unity editor.

We bring you concise, optimized APIs that work out of the box, and easy to integrate. The unified API lets you integrate once and works on all supported platforms!

New to Unity? Just don't want to modify your scripts, or just lazy? Don't worry! The CodelessScreenshotHelper extension is for you. Drag and drop the provided screenshot Prefab into your scene, even if the scene is already running in the editor. You are ready to capture images at any time you want!

The PLUS version enables the ability for adding watermarks, save and load on WebGL platform, and natively save your images to Android Gallery/iOS Photos, and more...

Features (Free)

- Capture full-screen image
- Capture part of the screen(Custom rect region at any screen position)
- Capture image using Unity camera, can scale the image size(0.1X – 4X)
- Support Unity built-in render pipeline and all Scriptable Render Pipelines
- Support Anti-Aliasing for camera capture methods
- Support touch, including touch-to-capture example
- Codeless screenshot extension for both runtime App and Editor
- Capture and returns Texture2D, Sprite, or RenderTexture
- Handy image display handler script for UGUI
- All platforms

Features + (Plus)

The below features are included in **Screenshot Helper Plus** only:
They provide important functionality to make the integrations easy and outstanding.

EasyIO

A powerful asset to save and load existing files, bytes data, and C# class objects in the common in-app paths. Supports all Unity player platforms, plus a WebGL plugin that provides extra support and function on WebGL.

- **Save** and **Load** images, file bytes (support custom extension name), C# class objects
- Works on all Unity player platforms (common in-app data paths of the platform)
- Support **WebGL** platform (persistent data path)
- Show the "**Save AS**" dialog box to save to local storage (**WebGL**)

Mobile Media Plugin

- Save Image, Video, GIF to native gallery
- Pick Image, Video, GIF from native gallery (single file picker)
- Save Audio to the Music folder (Android Only)
- Pick Audio from native folders (Android Only) (single file picker)
- Get thumbnail and full-size image for Image, Video, and GIF
- Save files to specific folder in the Gallery
- Save files with specific filename
- Options for picking file from Cloud drives (Android Only)
- Options for picking GIF as a static image(first frame) or origin file
- Check native storage permission and request permission
- Prompt native Settings menu for changing permission
- Handy external permission handler script and example
- Detailed example scene included, ready-to-build for testing on devices

*Demo scene included for showing how to pick/save image or video from/to mobile device gallery. Please find the Readme document and demo(**MobileMediaTest**.unity) in the MobileMedia folder.*

Watermark Util

- Add watermark texture on screenshot
- Add watermark texture on particular texture
- Support transparency
- Set watermark position
- Watermark grayscale mode option
- No need to place the watermark logo on UI
- Flexible, easy, texture combine API, can be used independently

Editor Screenshot Helper

- Capture any current selected editor window/tab using Hotkeys: **Shift + W**

(1) Reminders, Setup & Requirement

Scriptable Render Pipeline (SRP) : URP/LWRP/HDRP.

If you are using Scriptable Render Pipeline (e.g. URP/LWRP/HDRP) in your project, please ensure to use the **OnUpdateRender** mode for the camera capture methods. Just call the below API once before capturing images with camera:
ScreenshotHelper.SetRenderMode(ScreenshotHelper.RenderMode.OnUpdateRender);

Watermark Util (Plus Only)

Please set the Read/Write Enable flag as 'true' in the texture Import Settings for the imported textures and watermark icon.

CodelessScreenshotHelper for PLUS version (Plus Only)

(1) Save the captured images to **Android** Gallery/**iOS** Photos, by just ticking the native Save checkbox in the editor inspector. (by **MobileMedia Plugin**)
(2) Save the captures images in persistent data path on **WebGL**. (by **EasyIO**)
(3) Show the "Save As" dialog box to save to local storage on **WebGL**. (by **EasyIO**)

Mobile Media Plugin (Plus Only)

Requires **Android 4.4(API Level 19)** or later for Android platform.
Requires **iOS 8.0** or later for iOS platform.

For Android,
1. Set **Write Permission** to "External (SDCard)"
2. There are 3 Android aar plugin files included as below, select base on your need, unzip to replace the existing MobileMedia.aar.
   - **(a)** *MobileMedia-release-sdk29.zip* (default)
   - ***(b)*** *MobileMedia-release-sdk30+MANAGE_EXTERNAL_STORAGE.zip*
   - ***(c)*** *MobileMedia-release-sdk33+MANAGE_EXTERNAL_STORAGE.zip*
   - * Set **Target API Level to 29** in Unity Player Settings, for using aar **(a)**
   - * Set **Target API Level to 30 or newer** in Unity Player Settings, for using aar **(b)** or **(c)**

For iOS14 and later, "**All Photos**"(Read-Write) permission is required for saving files, and loading thumbnail & full-size image(without media picker) from Photos library.

(2) The MainOnCaptured callback

All capture methods in the ScreenshotHelper script will fire the **MainOnCaptured** callback on capture complete. It is like a central place for receiving captured images. So, add these code in your script for receiving all captured images if need:

** It is optional to set the MainOnCaptured callback, because every capture method has a callback for returning the captured image. We will talk about that later.

- **Set the callback to receive captured image in Texture2D format:**
```
ScreenshotHelper.iSetMainOnCapturedCallback((Texture2D texture2d)=>{
        //Your code for handling texture2d:
});
```
or
- **Set the callback to receive captured image in Sprite format:**
```
ScreenshotHelper.iSetMainOnCapturedCallback((Sprite sprite)=>{
        //Your code for handling sprite:
});
```
or
- **Set the callback to receive captured image in RenderTexture format:**
```
ScreenshotHelper.iSetMainOnCapturedCallback((RenderTexture renderTexture)=>{
        //Your code for handling renderTexture:
});
```

(3) Capture Methods

All screenshot capture methods has a callback for receiving the newly captured image. Capture a screenshot image can be done with just one line of code!

The callback parameter **onCapturedCallback** that used in the below examples can be a method name or Action, for receiving & handling the captured image:

- **Capture fullscreen image:**
```
ScreenshotHelper.iCaptureScreen(onCapturedCallback);
```

- **Capture a region of the screen:**

Specify a screen position and size(Vector2) to capture an particular area of the screen.
```
ScreenshotHelper.iCapture(Input.mousePosition, captureSize, onCapturedCallback);
```

- **Capture using the view of the specific camera:**
```
ScreenshotHelper.iCaptureWithCamera(camera, onCapturedCallback);
```
or
```
ScreenshotHelper.iCaptureRenderTextureWithCamera(camera, onCapturedCallback);
```

Anti-Aliasing

The anti-aliasing level can be set for better quality (smoother object edges) for camera capture methods when the **OnUpdateRender** mode is used, even if the Anti-Aliasing option is disabled in the Unity QualitySettings:
```
ScreenshotHelper.AntiAliasingLevel = 4;
```
Valid value: 1, 2, 4, 8 (1 = disable, 8 = best quality)
*To enable the **OnUpdateRender** mode, please use the below API:*
```
ScreenshotHelper.SetRenderMode(ScreenshotHelper.RenderMode.OnUpdateRender);
```

(4) Get the current texture/sprite

```
Texture2D texture2D = ScreenshotHelper.CurrentTexture;
Sprite sprite = ScreenshotHelper.CurrentSprite;
RenderTexture renderTexture = ScreenshotHelper.CurrentRenderTexture;
```

The above parameters stored the image of the previous capture. They will return a null if you never take a screenshot before. It is better to check null before using them.

(5) Save Image

Save the captured image using EasyIO, FileSaveUtil, FilePathName.

The **EasyIO** plugin (**Plus** only) :

```
SDev.EasyIO.SaveImage(texture2D, saveFormat, filename, folderName);
```

More methods of EasyIO for saving/loading different type of files, as below:

```
SaveImage, LoadImage, DeleteImage,
SaveBytes, LoadBytes, DeleteFile,
SaveString, LoadString, DeleteString,
SaveClassObject, LoadClassObject, DeleteClassObject,
WebGL_SaveToLocal, more...
```

The **FileSaveUtil** class:

```
SDev.FileSaveUtil.Instance.SaveTextureAsPNG(texture2D, specialFolder,
    subFolderName, filename);
```

FileSaveUtil allows save images/file bytes data to Mac, Windows special folders like Pictures, Music, Downloads, Documents, etc.

The **FilePathName** class:

```
FilePathName.Instance.SaveTextureAs(texture2D, saveFormat);
```

* The SaveFormat enum types can be JPG, PNG.

For saving image to the mobile device gallery please refer to the Readme document of MobileMedia Plugin (Plus Only).

(6) Clear

Call the Clear method to clean up memory if need, this methed will:

1) clear the stored textures
2) clear the main capture callback
3) remove the camera render script(CameraOnRender.cs/CameraOnUpdateRender.cs) from all cameras

```
ScreenshotHelper.iClear(bool: clearCallback, bool: clearTextures);
```

(7) CodelessScreenshotHelper

This is a code-less screenshot feature designed for capturing images without changing your codes. It provides a convenient way for saving your app/game screenshot with some simple setup in the scene. So you can capture images at any time when you need to.

1. Drag the prefab(**CodelessScreenshotHelper.**prefab) from the EditorExtension folder to your scene. Or, add the **CodelessScreenshotHelper**.cs script to a GameObject in the scene.
2. Set the capture settings in the inspector panel.
3. Start to capture images with the “Capture” buttons in the inspector panel.

Else, you can also reference the methods and dynamic parameters in the CodelessScreenshotHelper to your UI components like Button, Slider, InputField, and Toggle, etc. in the scene, this allows you to take screenshots at runtime in your app.

(8) Watermark Util

Watermark Util is one of our image processing tools, it can work independently for combining textures. Supports transparency and supports setting watermark position easily. Demo included, you can find the **WatermarkDemo** scene in the WatermarkUtil folder.

To add watermark on the captured screenshot or any existing texture:

Watermark method 1:

```
Texture2D newTexture = WatermarkUtil.DrawWatermark(
        targetTexture,  watermarkTexture,
        Vector2:watermarkPosition, float:watermarkAlpha,
        bool:drawInRect, bool:grayscale);
```

Watermark method 2:

```
Texture2D newTexture = WatermarkUtil.DrawWatermark(
        targetTexture, watermarkTexture,
        float:watermarkPositionX, float:watermarkPositionY, float:watermarkAlpha,
        float:watermarkWidthOffset, float:watermarkHeightOffset, bool:grayscale);
```

*Also, remember to set the Read/Write Enable flag as 'true' in the texture Import Settings for imported textures.

THANK YOU

Thank you for using this package!
For any question and bug report please contact us at swan.ob2@gmail.com.
Remember to rate this asset on the Asset Store. Your review is always appreciated, and very important to the development of this asset!

Review And Rating

Visit our asset page to find out more!
https://www.swanob2.com/assets

SWAN DEV