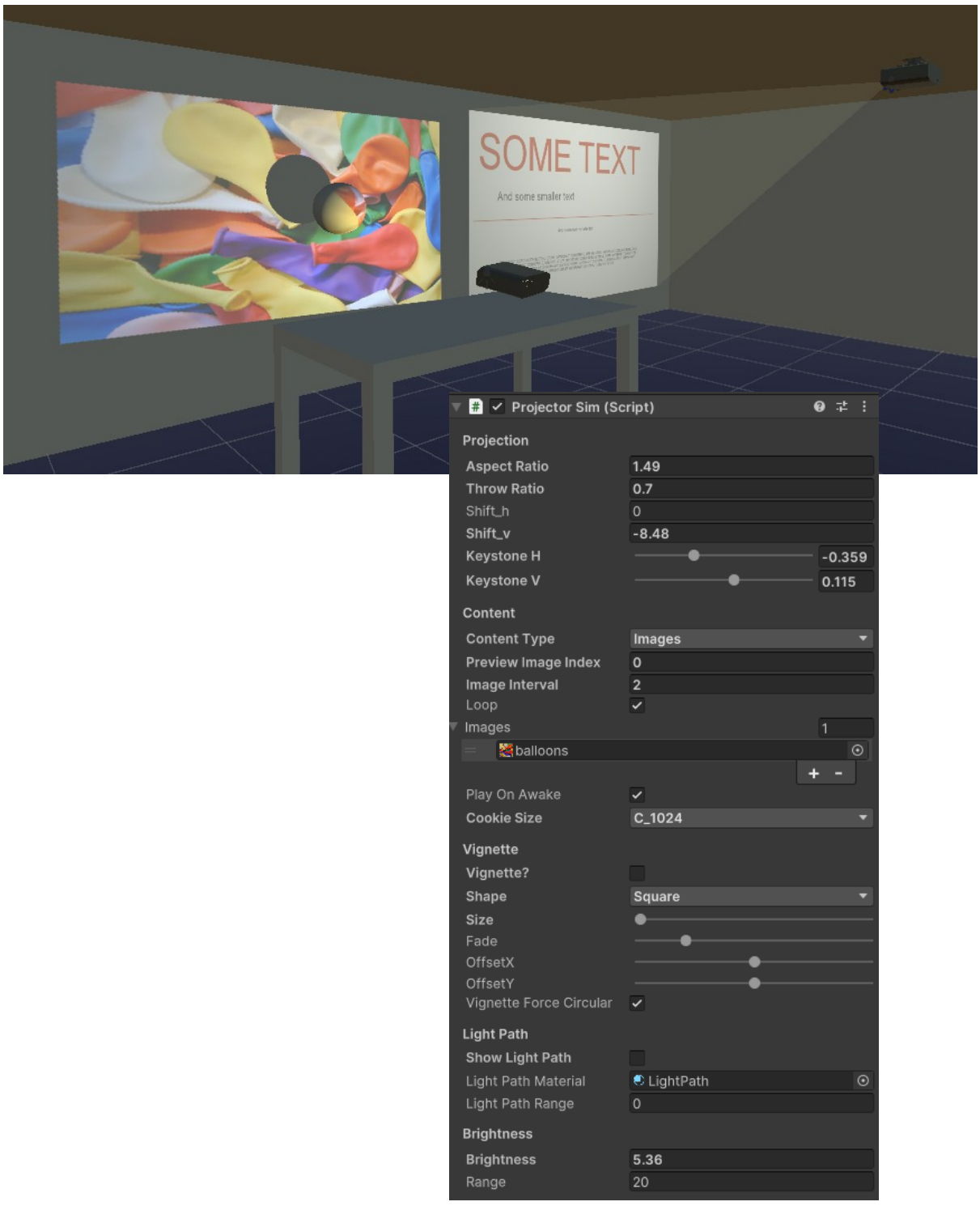
Projector Simulator

White Games

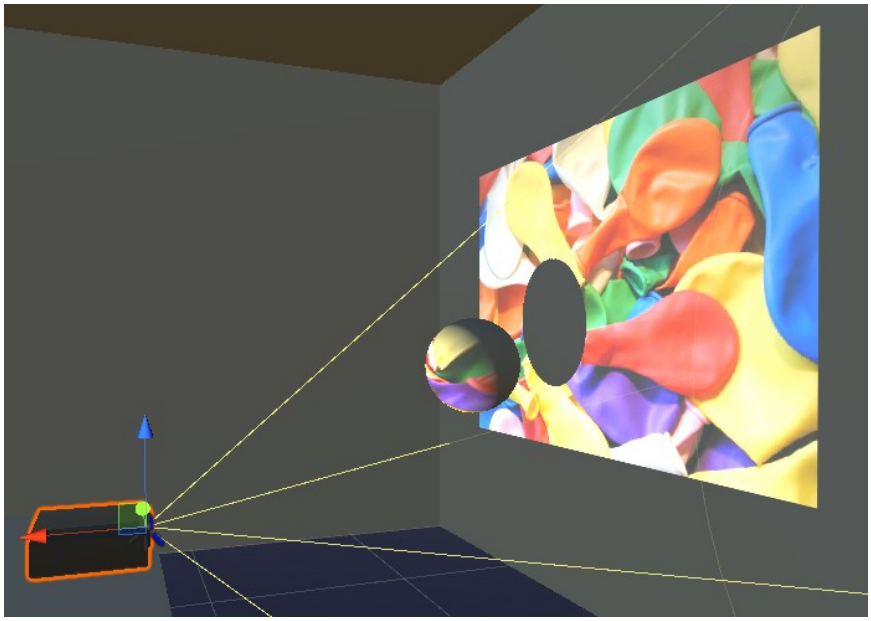


Web | Youtube | Twitter | Asset Store

Table of Contents

Overview

Projector Simulator is a Unity package which allows you to project any video, image, or series of images, in colour. It also simulates off-axis projection (lens shift) to create realistic shadowing effects and allowing projectors to be placed, for example, on a ceiling or table offset from the centre of the image.



Release Notes

V1.61

- Fixed console errors when selecting the Projector prefab in the Project tab

V1.6

- Code overhaul
- Custom inspector
- Simplified usage – now only one Projector prefab for all content types (Images/RenderTexture/Video)
- Removed CPU processing, now only shader-based processing
- Keystone adjustment now works in shader processing
- Improved organisation of files
- Documentation overhaul – removed outdated methods and references

V1.59

- Minor fixes on first time importing package
- Improved example scenes

V1.58

- Fix for URP + WebGL combination (still 2021.3 and later)
  - For WebGL you may need to delete all quality presets except for the highest

V1.57

- Fixed a bug that could cause crashes in builds when switching scene

V1.56

- Minor bug fixes (affects “standard” render pipeline only, not HDRP/URP)
  - Fixed a bug requiring projected textures to have read/write enabled, even when the shader-processing method is used (now only required for “legacy” CPU processing)
  - Fixed a bug that prevented RenderTexture/Video Projectors working when CPU processing used

V1.55

- Added URP support (2021.3 and later)
- Fixed minor memory leak when destroying projectors

V1.54

- Fixed brightness consistency issues in HDRP - projected image brightness should now reflect the lumens setting more accurately.

V1.53

- Added vignette effect
- Fixed HDRP projectors not being bright enough by default

V1.52

- Rewrote shader processing method
  - Shader method now supported from 2017.4 (previously 2018.3)
  - Shader method previously relied on a Graphics.CopyTexture call which is not supported on all platforms.
  - Now uses a custom single-pass shader which should improve performance and be supported on more platforms (e.g. WebGL)
  - “Border size” setting removed as no longer required with custom shader

  - Removed restrictions on Image and RenderTexture format requirements under the Shader method.

V1.51

- Minor bugfix in initial light colour in Unity versions prior to 2018.3.

V1.5

- Added support for HDRP (2018.4 and later)
  - HDRP package is included as a .unitypackage
  - Legacy processing method has been removed under HDRP, meaning keystone adjustment is no longer possible until it is added to the shader method

V1.42

- Minor bugfixes

V1.41

- Improved performance when projecting multiple RenderTextures simultaneously with the new Shader method (tested with 8 projectors)

V1.4

- Implemented new shader-based method, massively improving both performance and quality
  - Keystoning is not yet supported under this new method. Keystoning can still be used, but the projector will drop down to the “legacy” pre-1.4 pixel-by-pixel CPU-based method
  - You can also choose to force the legacy method to be used should you see any problems with the shader-based method

V1.33

- Made RenderTextureProjectors and VideoProjectors aware of each other's processing, so two projectors will not be processed in the same update in order to maintain a steady framerate. (disabled in 1.41 if using shader method)

V1.32

- Added light path geometry for simulating volumetric light
- Custom materials can be applied to the light paths
- Light paths do not currently take keystoning into account

V1.31

- Fixed a minor bug that prevented the new prefabs from working with default values
- Renamed the original “Projector” prefab to “ImageProjector” to be more in keeping with the new prefabs' naming conventions

V1.3

- Added ProjectorSim_RenderTexture script. This new type of projector is able to project RenderTextures (and by extension, video files) instantaneously, removing the need to manually extract a video's frames.
- Added RenderTextureProjector and VideoProjector prefabs

V1.23

- Added script to synchronise projected frames with an audio track (projected content will only progress when audio clip is playing - audio clip and projected content will have same length – see https://youtu.be/a1bsCE3JCPM)

V1.22

- Reduced cookie generation time by 10-30% depending on projector resolution/colour mode/number of images
- Memory optimisations

V1.21

- Added “Range” setting, as previously a projector's range was fixed at 10
- Increased default projector range to 20 instead of 10

V1.2

- Added keystone adjustment sliders
- Optimised generation of images after the first image in a projector with CPU method (subsequent images now use the first image as a starting point)
- Fixed a bug where the slideshow would play when calling “SetSlideshowIndex”, even if the slideshow was paused
- Other small optimisations/code cleaning

V1.1

- Added support for multiple images per projector
  - Now possible to project slideshows and quasi-video
- Added C# projector control interface
- Fixed a bug which caused images to be projected at the incorrect size – images now make the most use out of the available pixel space

V1.01

- Fixed a bug which caused all projectors to default to 256x256 resolution on level load

V1.0

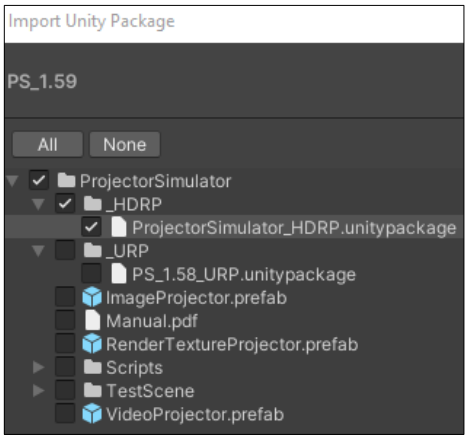
- Initial release

How to use

Getting started

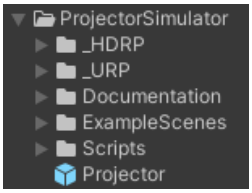
Note: When updating Projector Simulator to a new version, you should first delete the ProjectorSimulator folder before importing the new version.

HDRP/URP users: When importing the package, only import the .unitypackage in the _HDRP or _URP folder (overwriting the standard files may cause issues):



Once the .unitypackage is imported, you can unpack it. If you have already imported the standard files, delete the ProjectorSimulator folder and re-import the asset.

You will see something like this in the imported ProjectorSimulator folder:



The **Scripts** folder contains the 2 main scripts for the asset to work – *ProjectorSim.cs* and *CookieCreator.cs*.

It also contains other scripts:

- *ProjectorAudioSync.cs*, used for synchronising playback of projected Images with an audio track
- *ProjectVideo.cs*, which is used when projecting a video
- ThrowBuilder.cs, which builds light path geometry for a visible “light cone” effect

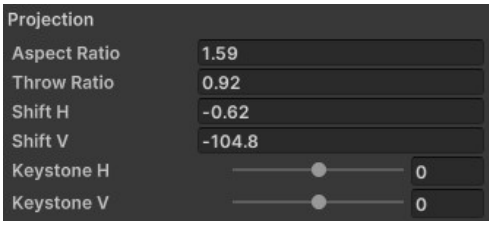
The **ExampleScenes** folder contains the example scenes and all the content used by them.

The **Documentation** folder contains this manual.

Finally, there is the **Projector** prefab which can be placed in your scene.

Positioning the image

1. To begin, drag and drop the Projector prefab into your scene
2. Position the projector so that the light source is in the desired location. Rotate the projector so that the forward direction is perpendicular to the display surface (if a rectangular image is desired)
3. Use the **Aspect Ratio**, **Throw Ratio**, **Shift_v**, **Shift_h, Keystone H and Keystone V** values to position the image in the desired location

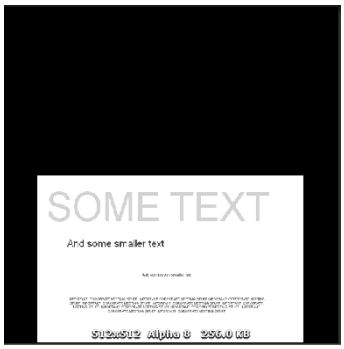


Aspect Ratio is the aspect ratio of the projected image (width divided by height).

Throw Ratio is equal to the projector distance divided by the image width, and is a standard measurement used by real-world projection lenses. Smaller values result in larger images. Must be greater than 0.

Shift H and **Shift V** correspond to the horizontal and vertical lens shift amount, respectively. Use this shift if you wish to achieve a square image with the light source above, below, or to the side of the image centre (e.g. a projector on a ceiling).

The **Keystone** sliders allow you to project a trapezoidal image. Usually used to make the image appear rectangular when your projector is not perpendicular to the screen surface. Note that keystoning can create unwanted artefacts in the projected image, as it is no longer an orthographic projection:



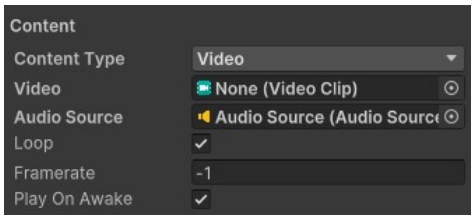
lens shift

keystone

You can also use the **Brightness** settings to adjust the brightness and reach of the image. Depending on the image size and projector distance, the image may appear too dull or too over-exposed.

Projecting content

4. Once the image is in place, use the **Content** controls to set what the projector will display



The **Images** parameter is where you can drag and drop your image files. Several images can be added to the Images list to create a slideshow.

The **Preview Image Index** value is used to control which image is shown when the editor is in edit mode.

Image Interval is the amount of time in seconds to show each image, if more than one has been added.

The **Loop** option allows the array of images to be cycled through continuously. Unchecking this option causes the projector to freeze on the last image when is it reached.

Note: Previous versions of Projector Simulator processed all the images into precalculated cookies at the start of the level. Now, each image is processed in realtime as it is needed by blitting the required image onto a RenderTexture, which is then projected. The RenderTexture is created at the start of the level, and is the same resolution as the largest Image in the list.

If projecting a RenderTexture, you can select the RenderTexture that you wish to project.

When projecting a RenderTexture or Video, you see a **Framerate** setting. If this value is negative, the projected image will be updated every frame. Positive values will update the projected image at the given rate per second.

The **Cookie Size** setting sets the size of the cookies used by the projector's light sources. This should remain at a lower setting (e.g. C_512 or C_1024) while the image is being positioned, otherwise stuttering may occur during step 3. Once the image is in place, you can experiment with this property until a suitable image quality is achieved. Higher resolutions will take longer to update the projected images which may result in stutter, but as everything is now processed in shaders this "shouldn't" be much of an issue.

Play On Awake has different behaviours depending on the content type. It is tied to the public boolean variable *playOnAwake*.

The following table describes the effects of the **Play On Awake** setting at the start of the level, and the effect of calling the projector's other functions during gameplay.

	Content type		
	Images	RenderTexture	Video
Play On Awake unticked	Pause on first image	Projector turned off	Projector turned off and video doesn't play
Pause()	Pauses on current image (sets playOnAwake to false)		
TogglePause()	Pauses image or resumes playback.		
Play()	Resumes playback. Will also enable the ProjectorSim component (turn the projector on) if it is disabled		
Disable/Enable ProjectorSim component	Turns projector off/on.		Turns projector off/on Disabling does NOT pause the video (audio may continue) – call Pause() when disabling if you want the video to pause Instead of enabling, call Play() to enable the component and start playback immediately.

See the ProjectorControl example scene for an example of how to control a Projector via script.

To start the level with the projector off completely, disable the ProjectorSim component. Enable the component via script when you want to the projector to turn on.

Playback Control

There are a few functions included to enable your own scripts to control the playback of the ImageProjector. Examples are included in the *ProjectorControl* scene and *ProjectorControl.cs* script.

```
ProjectorSim pj;
```

To turn the projector on and off, simply enable/disable the ProjectorSim component.

```
pj.enabled = true;
pj.enabled = false;
```

To pause or resume the playback, call the functions:

```
pj.Pause();
pj.Play();
pj.TogglePause();
```

You can manually force the Image slideshow to advance to the next frame with the function:

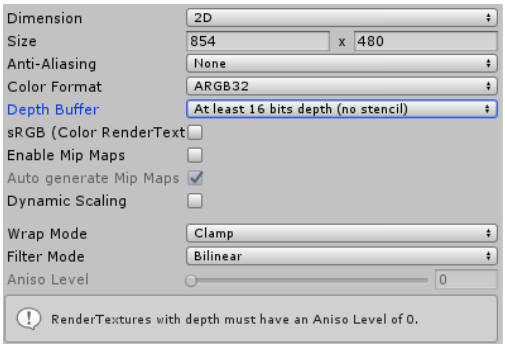
```
pj.AdvanceSlideshow();
```

Or you can make an Image projector jump to a specific frame:

```
pj.SetSlideshowIndex(5);
```

Content Import Settings

RenderTextures

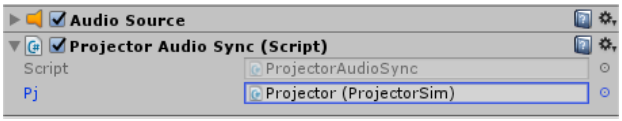


If you are projecting a RenderTexture that is rendered by a camera, you should include a depth buffer in the RenderTexture settings.

Synchronising projected Images with an audio track

Note: *consider projecting a video instead.*

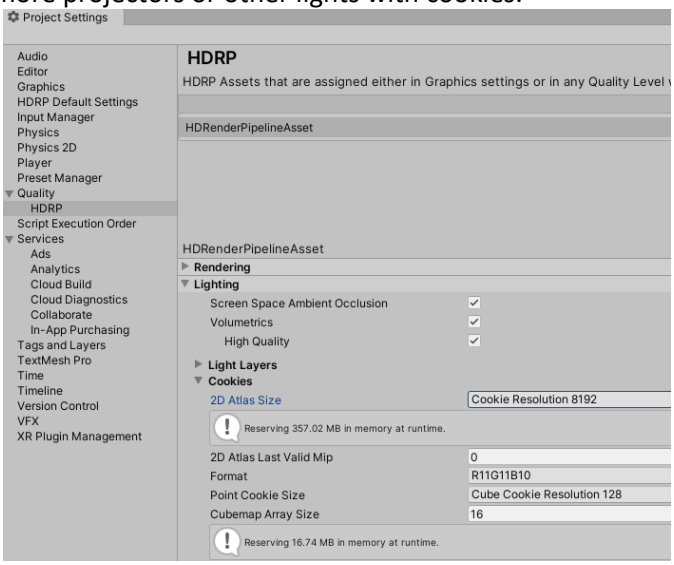
A script is provided named *ProjectorAudioSync.cs*. This script can be added to an object with an Audio Source component, and then given a reference to the Projector with Images you want to have synchronised to the audio:



You should turn off “Play On Awake” on the Projector.
This will cause the Projector to play through its Images at a constant rate whenever the audio is playing, such that the projected slideshow and the audio clip will have the same length. If you want to pause the slideshow, just pause the audio.
An example of this can be seen in this video: https://youtu.be/a1bsCE3JCPM

HDRP settings

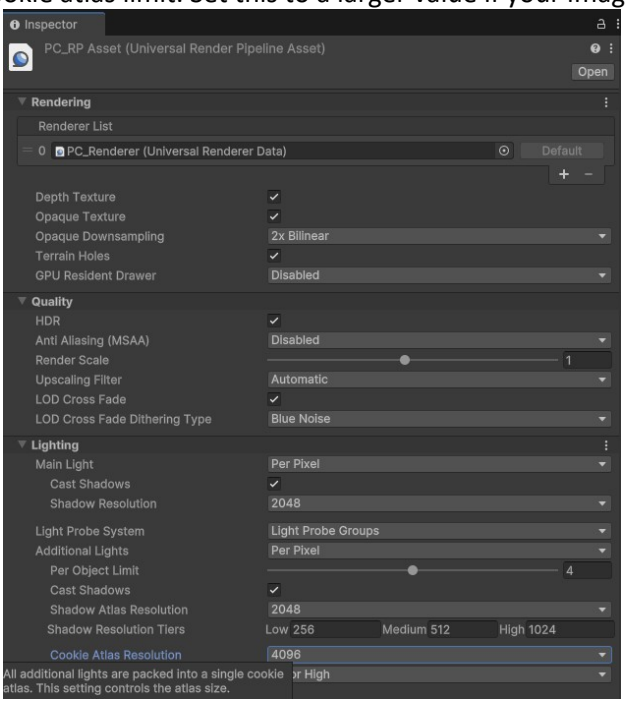
When you load the HDRP example scene, you may find that the images are quite blurry or you get a warning about the cookie atlas. This is because of HDRP's low default cookie atlas resolution. You should increase this setting until you have acceptable image quality. It may have to be increased again later if you add more projectors or other lights with cookies.



The projectors may appear to have a very low brightness to begin with. Increase the brightness on the projector script until the brightness is acceptable (can be up to 100 000 depending on what seems to be camera exposure).

URP settings

URP also has a similar cookie atlas limit. Set this to a larger value if your images are blurry.



Known Issues/Troubleshooting

1. Occasionally, the projector may stop responding to updates in the editor's Inspector.
    - The cause of this is unclear, but it seems to only sometimes occur on project load, or when Unity has been in the background for an extended period of time, or when scripts are compiled.
    - To rectify the issue, simply play and stop your scene.
    - This issue should be fixed in version 1.6, please contact us if you see this issue occuring.
2. The projector(s) introduce a strange coloured lighting effect on the projected image, or the rest of the scene (built-in render pipeline only, not HDRP/URP)
    - This is likely caused by the **Pixel Light Count** being set too low – each colour projector adds 3 spotlights to your scene, and when using Forward Rendering the pixel light count setting may need to be increased.
    - The **Pixel Light Count** setting can be found in the Quality settings. An example of this effect and fix can be seen in our Introduction and Tutorial video.
    - You should ensure your Pixel Light Count is high enough in all of your possible quality settings, as some users may choose/be forced to run at a lower preset.
    - Alternatively, use the Deferred renderer, which is recommended when using many dynamic light sources.
3. A projected image appears in the Game view, but not in the Scene view
    - As Projector Simulator makes use of real Unity spotlights, ensure you have lighting enabled in the Scene view.

Authenticity Statement

If you downloaded or purchased this asset from anywhere other than the Unity Asset Store, then unfortunately you have a pirated version of Projector Simulator. Please support the original creator by purchasing directly from the Unity Asset Store.
Alternatively, show your support by buying me a coffee: https://ko-fi.com/shwhjw

如果您从 Unity Asset Store 以外的任何地方下载或购买了此资产，那么很遗憾，您拥有的是盗版的 Projector Simulator。请直接从 Unity Asset Store 购买以支持原创者。
或者，您也可以给我买杯咖啡以表达您的支持：https://ko-fi.com/shwhjw