

# Smart Lighting 2D

Updated: 2019 / 02 / 28

Current Asset Version: [1.0.6](#)

Latest Documentation: [Link](#)

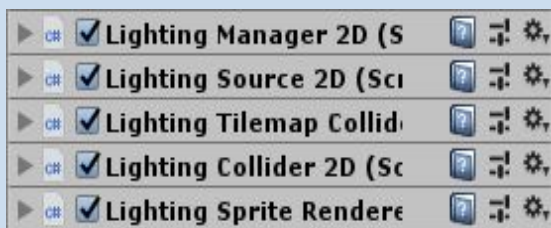
Roadmap: [Link](#)

Forum Discussion: [Link](#)

Mail: [simonas@kuzmickas.lt](mailto:simonas@kuzmickas.lt)

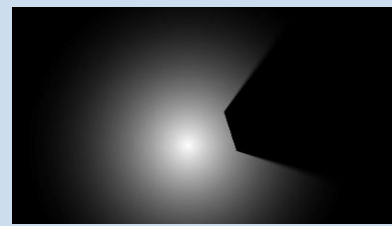
Discord Support: [Link](#) 

## User Manual Sections



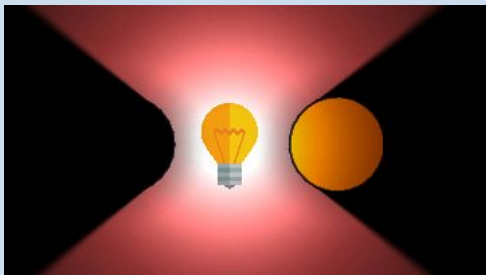
### Component Reference

A complete description of each component of 2D Lighting System.



### How to Start?

A basic sample of how to set up lighting system for a new scene.



### What is Light Masking?

Explanation and Samples on how masking can be used to achieve certain results



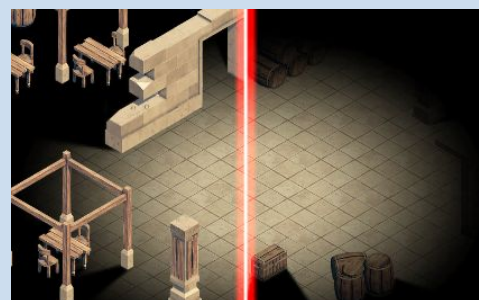
### Custom Physics Shape

What is Sprite's Custom Physics Shape?  
How can we use it?



### Super Tilemap Editor Support

Having issues with Super Tilemap Support?



### Lighting Sorting Layer

Having issues with sorting layer & order?

**If there are any questions/feature requests or need help, feel free to contact!**

**I will answer the questions before you decide to use the asset.**

**Ask or find more information in [Discord!](#)**

**Documentation is the development, there will be updates!**

# Unity Lighting 2D Components

## Components

<a href="#">LightingManager2D</a>	Component is used to setup lighting settings, enable optimizations. Only one Lighting Manager 2D is allowed per scene. This component is <b>automatically generated</b> on first 2D lighting API call.
<a href="#">LightingSource2D</a>	Versatile light emitting source, this component emits light. Can use custom texture to setup unique light emitting look.
<a href="#">LightingCollider2D</a>	Versatile lighting collider component. Can be used to set up shadow casting for sprites & solo collider components.
<a href="#">LightingTilemapCollider2D</a>	Tilemap Lighting Collider component can be used to setup shadow casting for standard unity Tilemap component and <b>Super Tilemap Editor</b> system.
<a href="#">LightingSpriteRenderer2D</a>	Lighting Sprite Renderer component can draw images with different blending modes straight into lighting buffer to light up objects, particles and other scene entities. This component is very efficient to make lights without shadow casting. <b>(Very Mobile Friendly)</b>
<a href="#">LightingRoom2D</a>	This component can be used to darken the area in the daylight. For example you might need to have a dark room in brightly lit scene with daylighting shadows.
LightingTilemapRoom2D	This component can be used to darken the area in the daylight. Can be used to mask tilemap to be affected by lights.

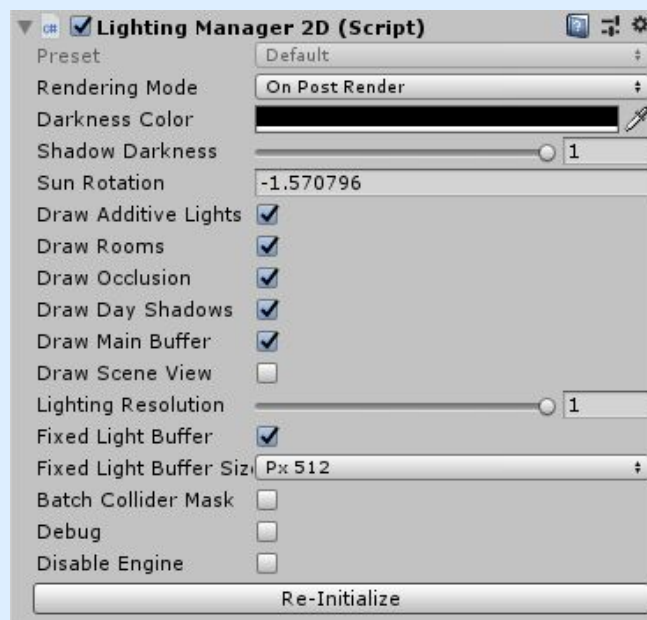
## Bonus Components

<a href="#">ColliderLineRenderer2D</a>	Creates an outline for Collider components with selected color
<a href="#">Mesh2D</a>	Creates a mesh from Collider components and attach it to mesh renderer. Mostly used to display basic demo scenes without any images.

# Component Reference

## Lighting Manager 2D

appearance



description

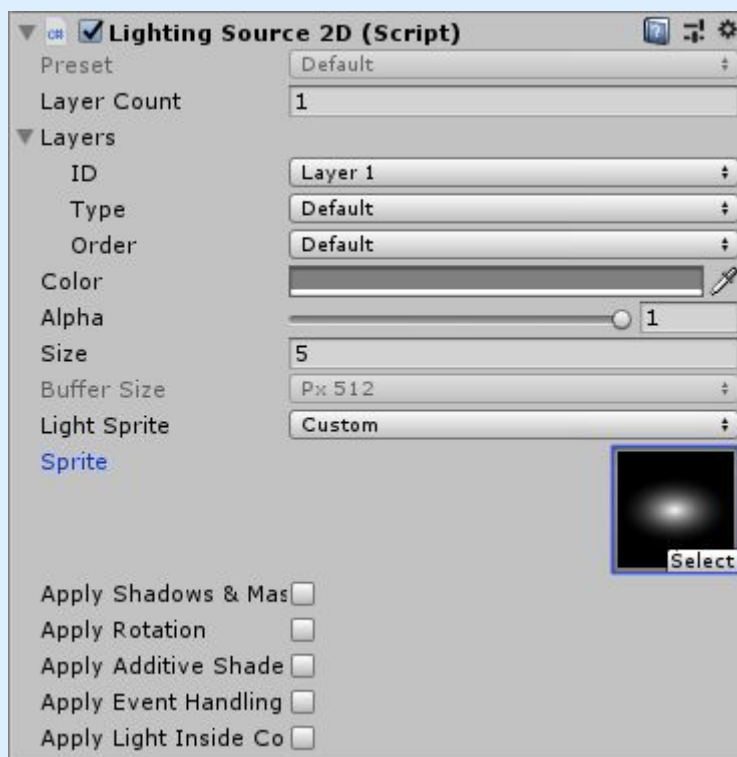
*Component is used to setup lighting settings, enable optimizations. Only one Lighting Manager 2D is allowed per scene. This component is automatically generated on first 2D lighting API call.*

information

	Rendering Mode	On Render	Additional Sorting Order option will appear for this setting. You can set specific sorting order for lighting buffer.
		Pre Render	Game objects with sorting order will appear above the lighting buffer. Game objects with lower sorting order ID will appear below the lighting buffer.
		Post Render	Not recommended to use. Lighting buffer is drawn on post process rendering loop. It seems to have many issues if you want to use post-processing effects.
	Darkness Color	24 Bit Color	The darker color, the darker scene will be drawn. For day lighting effects you should set up darkness color very bright. Can be used to get "tint" which could represent dusk or dawn.
	Shadow Darkness	Float [0 - 1]	The darkness of day lighting shadows. 0 - not visible, 1 - opaque.
	Sun Rotation	Radians	Sun rotation will affect all lighting colliders with day shadows. This variable can be manipulated in real time to achieve day & night cycle.
	Draw Additive Lights	Boolean	Enable additive lights drawing. When disabled, it will skip all checks for additive lights drawing. If you are not using this feature in any of the lights, it is suggested to disable this.
	Draw Rooms	Boolean	Enable rooms feature, mostly used for scenes with day lighting effects.
	Draw Occlusion	Boolean	Enable occlusion drawing, currently not recommended to use, this feature is going to be improved in 1.0.6 and 1.0.7 together with day lighting.
	Draw Day Shadows	Boolean	Enable day shadow casting for lighting colliders. It is recommended to disable this if you are not using day lighting effects.

	Draw Main Buffer	Boolean	<i>When disabled, it hides lighting buffer from main camera. This setting is similar to "Disable Engine" except all lighting calculations will be still performed, but not drawn.</i>
	Draw Scene Buffer	Boolean	<i>When enabled, lighting sources can be seen in scene view.</i>
	Lighting Resolution	Float [0.125 - 1]	<i>The resolution of Lighting buffer. The higher resolution is, the more detail lighting is, however it also impacts the performance. For very high resolution, it's recommended to reduce lighting resolution because additional crispy details are not very noticeable in higher resolutions than 720x1280. It is recommended to set this setting 0.5 for mobile device build.</i>
	Fixed Light Buffer & Size	Boolean Enumerator	<i>This option enables better poll system for light source system. Improves performance and is recommended for mobile build. When enabled, custom light buffer size is not available, all lights will be having same texture size.</i>
	Batch Collider Mask	Bool	<i>When enabled, lighting will use same texture as a source for sprite masks. It won't work unless all sprites are included in the same texture. Use this to improve performance.</i>
	Debug	Bool	<i>When enabled, additional lighting information will be displayed in game view. This helps to benchmark lighting performance. Mostly used by Smart Lighting 2D Developer.</i>
	Disable Engine	Bool	<i>This option disables all lighting features, no lighting calculations will be applied.</i>

## appearance



## description

Versatile light emitting source, this component emits light. Can use custom texture to setup unique light look.

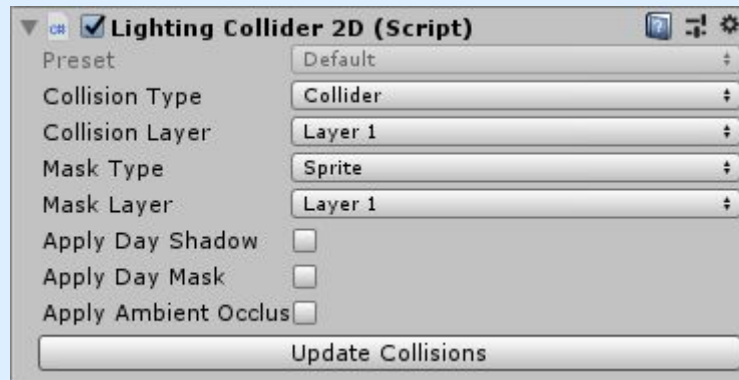
## information

Layer Count	Int [0 - 31]	Layer count that will be included in light calculations. The more layers, the less optimized light calculation will be. 3 layers should be enough for making quite complicated or tricky lighting scenes.
Layers	Layer Object List	The list of layer objects. For each layer select it's <b>Id</b> that is used in every lighting collider. <b>Type</b> gives an option to draw colliders or masks only. <b>Order</b> allows you to sort the shadows and masks according to specific statement (Distance To Light, Y Axis)
Color	24 Bit Color	The color of light. The darker the color, the less visible it will appear. Black color is not visible at all.
Alpha	Float [0 - 1]	Transparency of light. The higher alpha value, the more visible light appears to be.
Size	Int [0 - Unidentified]	The size of light, keep in mind that increasing the size of light does not automatically increase it's "buffer size", very large lights requires larger buffer size, otherwise pixelated artifacts appear.
Buffer Size	Enumerator	The resolution of light buffer. Larger buffer leads to more crispy shadow details, however it costs more performance.
Light Sprite	Default	Default texture which is being applied for the light.
	Custom	Enables custom sprite texture to use for the light.
Sprite	Sprite	After enabling LightSprite/Custom, you are able select your custom sprite texture for the light.
Apply Shadows & Masks	Boolean	An option to disable shadow and mask drawing. However if you do not need these features, it is recommended to use Lighting Sprite Renderer instead.
Apply Rotation	Boolean	Enable object transform rotation for the light

	Apply Additive Shader	Boolean	<i>Enable additive shader for the light</i>
	Apply Event Handling	Boolean	<i>Enable ability to use event handling feature with this particular lighting object.</i>
	Apply Light Inside Collider	Boolean	<i>By default, once light appears in the collider, no collisions are generated with that particular object. Once this option is enabled, you can put light inside the objects and light will still collide with their walls.</i>

## Lighting Collider 2D

appearance



description

*Versatile lighting collider component. Can be used to set up shadow casting for sprites & solo collider components.*

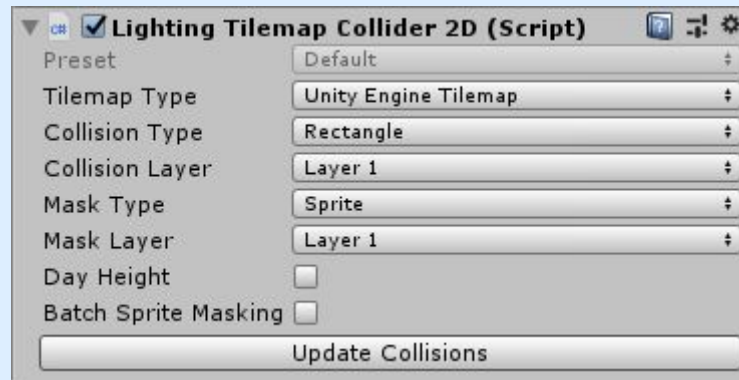
information

Collision Type	None	<i>Disables all shadow casting for this object.</i>
	Collider	<i>Use 2D collider for geometry of shadow casting. (Box2D, Circle2D, Capsule2D, Polygon2D, Edge2D)</i>
	Sprite Custom Physics Shape	<i>Use Sprite's custom physics shape which can be accessed with Unity Sprite Editor.</i>
	Mesh	<i>Uses Mesh Filter Mesh to cast shadows.</i>
Collision Layer	Layer Enumerator [0 - 31]	<i>Lighting layer of the object, this layer should be included in the lighting source layer list.</i>
Mask Type	None	<i>Disables all masking for this object.</i>
	Sprite	<i>Uses sprite from sprite renderer of this object for the mask.</i>
	Collider	<i>Uses 2D Collider geometry for the mask.</i>
	Sprite Custom Physics Shape	<i>Use Sprite Custom Physics Shape geometry to mask the object.</i>
Mask Layer	Layer Enumerator [0 - 31]	<i>Lighting layer of the object, this layer should be included in the lighting source layer list.</i>
Apply Day Shadows	Boolean	<i>Enable day lighting for the object</i>
Apply Day Mask	Boolean	<i>Enable day masking for the object. Day masking is used to avoid shadow casting on itself.</i>
Apply Ambient Occlusion	Boolean	<i>Enable ambient occlusion.</i>
Update Collisions	Editor Button	<i>Press this object to re-initialize geometry of the collider. This is workaround for performance reasons because geometry is not updated in real time. For example this should be triggered after changing polygon collider geometry (editor run time). Keep in mind that after going into play mode everything is applied automatically.;</i>



## Lighting Tilemap Collider 2D

appearance



description

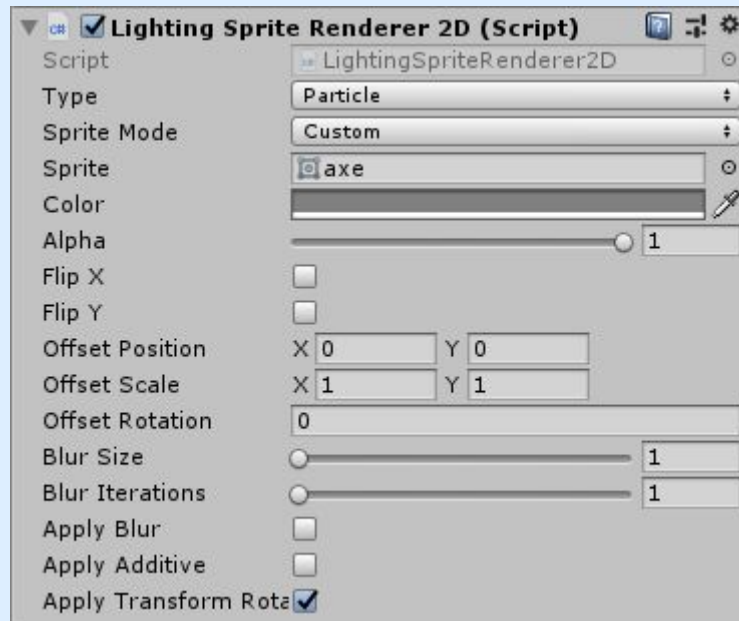
*Tilemap Lighting Collider component can be used to setup shadow casting for standard unity Tilemap component and [Super Tilemap Editor](#) system.*

information

Tilemap Type	Unity Engine Tilemap	Use standard tilemap for shadow casting. No additional collider components are needed for this. Lighting system will take sprites used in the tileset and apply their selected properties for collisions.
	Super Tilemap Editor	Use <a href="#">Super Tilemap Editor</a> for the shadow casting.
Collision Type	None	Disable shadow casting for this tilemap object.
	Rectangle	All tiles are treated like rectangles
	Sprite Custom Physics Shape	Try using sprite's custom physics shape for shadow casting.
Collision Layer	Layer Enumerator [0 - 31]	Lighting layer of the object, this layer should be included in the lighting source layer list.
Mask Type	None	Disable masking for this object.
	Sprite	Uses tile sprites for masking.
	Rectangle	Use rectangle shape for masking for masking this tilemap.
	Sprite Custom Physics Shape	Use tile sprite custom physics shape.
Mask Layer	Layer Enumerator [0 - 31]	Lighting layer of the object, this layer should be included in the lighting source layer list.
Day Height & Size	Enumerator Float [0 - Undenified]	Enable sun's shadow casting in daylighting system.
Batch Sprite Masking	Boolean	This is optimization. Enable this option when whole tile palette consist of same texture file. This should improve batch calls from lighting system.
Update Collisions	Editor Button	Press this object to re-initialize geometry of the collider. This is workaround for performance reasons because geometry is not updated in real time. For example this should be triggered after changing polygon collider geometry (editor run time). Keep in mind that after going into play mode everything is applied automatically.;



## appearance




## description

*Lighting Sprite Renderer component can draw images with different blending modes straight into lighting buffer to light up objects, particles and other scene entities. This component is very efficient to make lights without shadow casting. (**Very Mobile Friendly**)*


## information

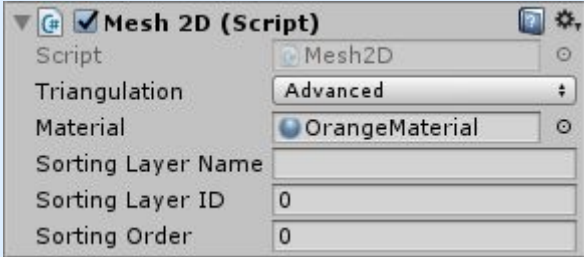
Type	Particle	Additive shader effect for this component.
	White Mask	Applies white mask for this object, the object is always fully visible and over the lighting buffer. However, this can be also achieved using sorting order which is higher than lighting buffer sorting order.
	Black Mask	Applies black mask for the object, object and everything underneath is completely not visible.
Sprite Mode	Custom	Select your own sprite for this component.
	Sprite Renderer	Synchronize sprite variable with Sprite Renderer component attached to the same game object.
Color	24 Bit Color	Color of the effect. This is not taking any effect when using white mask or black mask.
Alpha	Float [0 - 1]	Transparency of this effect. This is not taking any effect when using white mask or black mask.
Flip X	Boolean	Flips the sprite on the X axis.
Flip Y	Boolean	Flips the sprite on the Y axis.
Offset Position	Vector 2	Offset sprite's position.
Offset Scale	Vector 2	Additional scale offset for the sprite.
Offset Rotation	Degrees	Additional rotational offset for the sprite.
Blur Size	Int [1 - 10]	When blur is enabled, you may choose it's strength.
Blur Iterations	Int [1 - 10]	The times blur algorithm is being applied.
Apply Blur	Boolean	For this option to be used, you need to enable sprite write/read setting.

	<i>Apply Additive</i>	Boolean	<i>Apply additive shader for the lighting sprite renderer.</i>
	<i>Apply Transform Rotation</i>	Boolean	<i>Enable transform offset for the sprite.</i>

Lighting Room 2D			
appearance			
description	<i>This component can be used to darken the area in the daylight. For example you might need to have a dark room in brightly lit scene with daylighting shadows.</i>		
information	Color	24 Bit Color	<i>The color of the room</i>

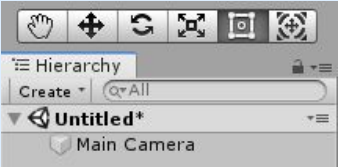

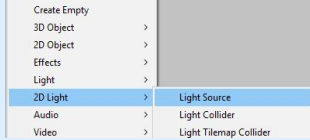
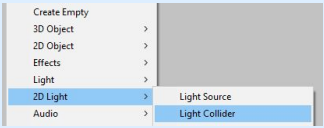
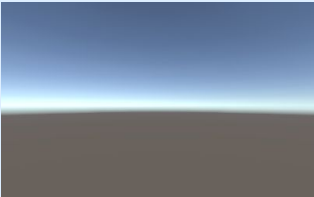
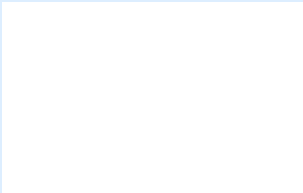

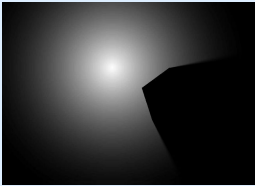
# Bonus Component Reference

ColliderLineRenderer2D		
appearance		
description	<i>Draws lines for attached Collider2D</i>	
information	<i>Color</i>	<i>The color of line</i>
	<i>Line Width</i>	<i>The width of line</i>

Mesh2D		
appearance		
description	<i>Creates a mesh from already attached Collider2D</i>	
information	<i>Triangulation</i>	<i>Triangulation method to be used when generating 2D mesh.</i>
	<i>Material</i>	<i>The material of the object.</i>

# How to Start?

Instructions		
Step 1	Creating a new Scene	Create a new scene in tab <b>“File/New Scene”</b> .
Step 2	Camera Setup	Make sure to have orthographic mode set for the camera. Set scene background to be quite bright. If you'll have black background, your default setup lights & shadows won't be visible.
Step 3	Creating a Light Source	Create a light in tab <b>“GameObject/2D Light/Light Source”</b> .
Step 4	Creating Light Manager	After creating the light, Lighting Manager should be <b>generated automatically</b> . At this step, you should not do anything, except to check if Lighting Manager 2D is in the root of hierarchy. If not, try start/stop the scene.
Step 5	Creating Light Collider	Create a collider in tab <b>“GameObject/2D Light/Light Collider”</b> After adding this object into the scene, you should be able to see light collisions with the collider. The shadow should be visible, collider object should be black. For making collider visuals visible read more in <b>“<a href="#">What is Masking</a>”</b> .

Step 1	Step 2	Step 3	Step 4
Visual Explanation			
			
Game View			
			

# What is Masking?

## Introduction

*Masking feature allows your objects to appear above the shadows.*

## Instructions

Step 1	Setting Up Scene & Camera	Create and Setup a new scene for this sample. Do not forget to use orthographic camera and white background for the scene.
Step 2	Creating a Light Source	Create a light in tab <b>"GameObject/2D Light/Light Source"</b> .
Step 3	Creating a Sprite	Creating a new <b>"GameObject"</b> and attaching <b>"Sprite Renderer"</b> component to it.
Step 4	Attach Light Collider To Sprite	Attaching <b>"LightingCollider2D"</b> component to already existing <b>"GameObject"</b> with sprite.
Step 5	Setup Lighting Collider	Make sure Mask Type is <b>"Sprite"</b> . So the shape of <b>"Sprite Renderer"</b> sprite will be masked and visible for the light source.  Also make sure Collider Type is <b>"Sprite Physics Shape"</b> . In that case you don't need to attach any collider components for object to cast shadows.

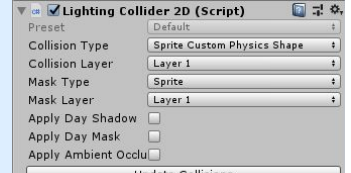
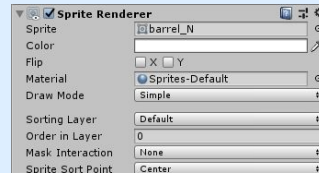
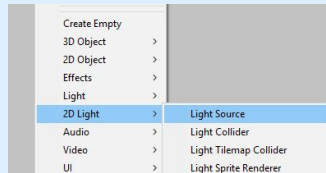
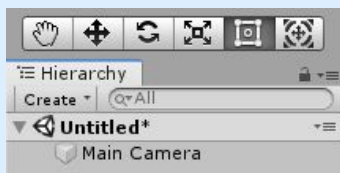
Step 1

Step 2

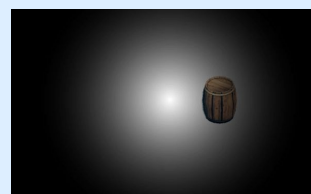
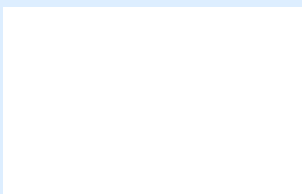
Step 3

Step 4 & 5

## Visual Explanation



## Game View



# Custom Physics Shape

## Introduction

The Sprite Editor's Custom Physics Shape allows you to edit a Sprite's Physics Shape. You can use this specific information from the sprite to cast shadows instead of using Collider component attached.

Unity Documentation: [Link](#)

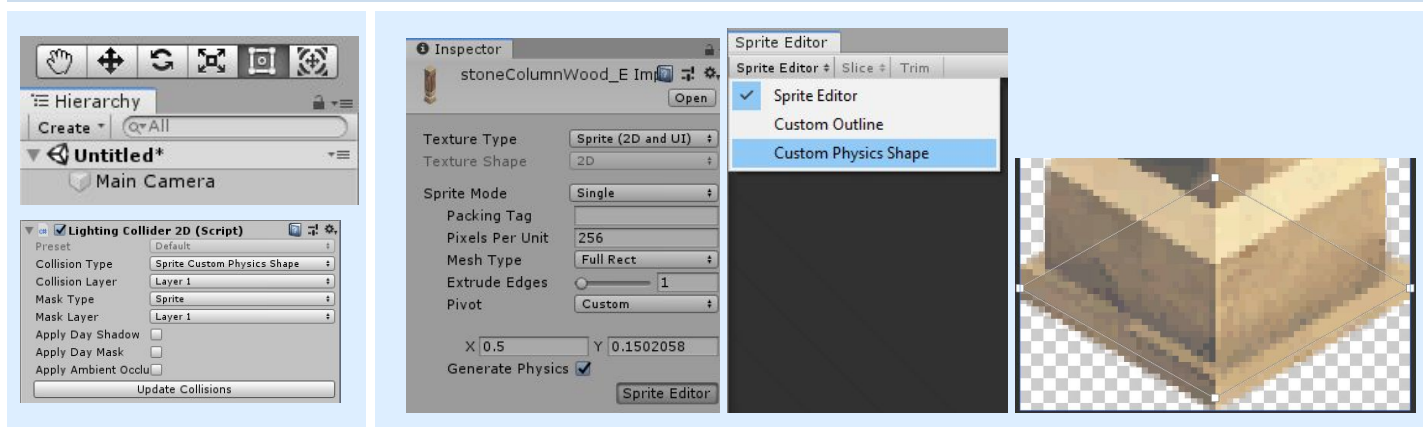
## Instructions

Step 1	Setting Up Scene & Camera	Create and Setup a new scene for this sample. Do not forget to use orthographic camera and white background for the scene.
Step 2	Creating a Light Source	Create a light in tab "GameObject/2D Light/Light Source".
Step 3	Creating a Sprite	Creating a new "GameObject" and attaching "Sprite Renderer" component to it.
Step 4	Attach Lighting Collider	Attaching " <b>LightingCollider2D</b> " component to already existing "GameObject" with sprite. Make sure <b>Mask Type</b> is " <b>Sprite</b> ", so the shape of " <b>Sprite Renderer</b> " sprite will be masked and visible for the light source. Also make sure you are using <b>Collider Type</b> " <b>Sprite Physics Shape</b> ".
Step 5	Setup Custom Physics Shape	Go to the <b>Sprite Import Inspector</b> and press " <b>Sprite Editor</b> " button. Then switch to <b>Custom Physics Shape</b> mode. There you can add and edit vertices of shadow casting collider. Do not forget to press " <b>Apply</b> " after finishing to edit the shape.

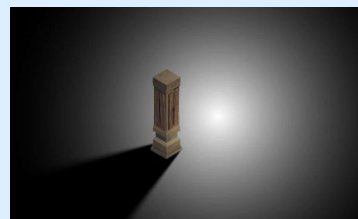
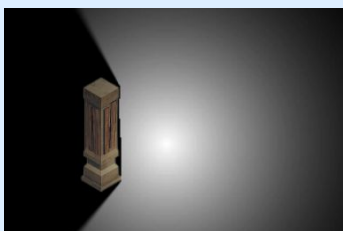
## Step 1 & 2 & 3 & 4

## Step 5

## Visual Explanation



## Game View



# Super Tilemap Editor Support

## Introduction

*Super Tilemap Editor is a powerful and easy to use tile editor with everything you need to create any game based on tiles. Use it not only to create tilemaps but also as a powerful level editor placing prefabs as if they were tiles.*

[Asset Store Link](#)

## How To Enable Support?

Step 1	Open <i>LightingTilemapCollider2D.cs</i>	The file is located in <b>“Assets\FunkyCode\SmartLighting2D\Components”</b>
Step 2	Uncomment Source Code	<pre>// Uncomment The Code Under The Line case MapType.SuperTilemapEditor:</pre>
Step 3	Enjoy SuperTilemap Support	:)



# Lighting Sorting Layer

## Introduction

*Sorting Layers and Order in Layer are used to determine the render order of lighting buffer in a scene.*

Unity Documentation: [Link](#)

## Instructions

Step 1	Setting Up Scene & Camera	Create and Setup a new scene for this sample. Do not forget to use orthographic camera and white background for the scene.
Step 2	Creating a Light Source	Create a light in tab <b>"GameObject/2D Light/Light Source"</b> .
Step 3	Creating a Sprite	Creating a new <b>"GameObject"</b> and attaching <b>"Sprite Renderer"</b> component to it.
Step 4	Create Sorting Layers	Create a new layers in tab <b>"Edit/Project Settings/Tags and Layers"</b> . Call the first layer <b>"My Sprites"</b> Call the second layer <b>"My Lighting"</b>
Step 5	Assign Sorting Layer To Sprite	Go to the object with Sprite Renderer, apply "My Sprite" layer in "Sorting Layer" dropdown menu.
Step 6	Assign Sorting Layer To Lighting	Go to "Lighting Manager 2D" GameObject which should have been automatically generated in the scene. Set Sorting Layer Name to <b>"My Lighting"</b>

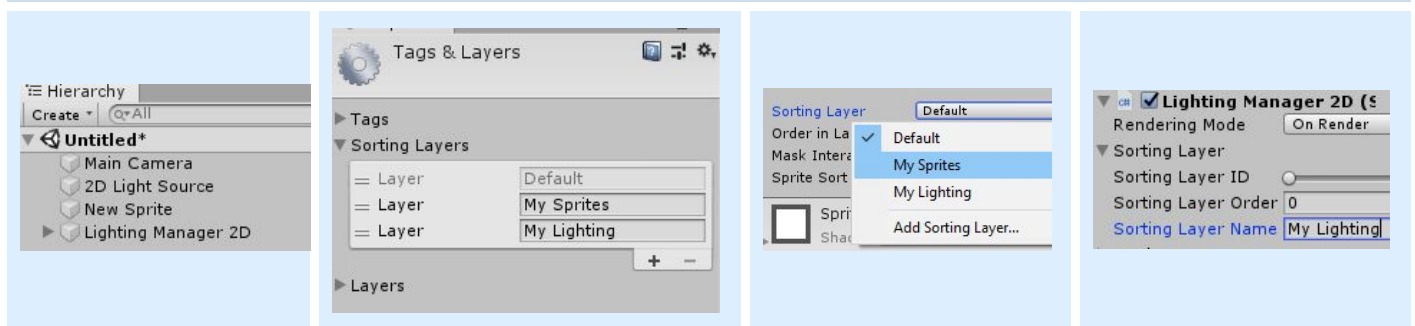
Step 1 & 2 & 3

Step 4

Step 5

Step 6

## Visual Explanation



## Game View

