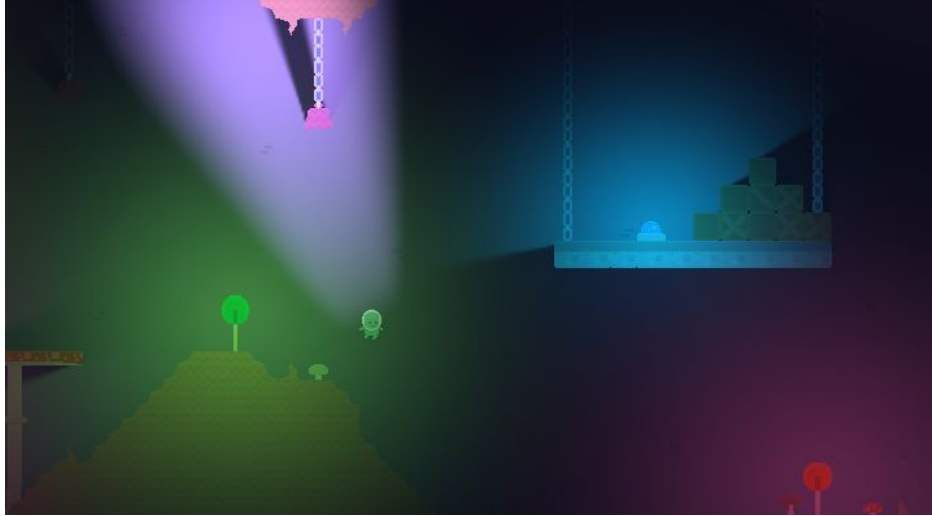


# Super Fast Soft Shadows



Adding support for Super Fast Soft Shadows to your project consists of four easy steps. First, you need to attach a **SFRenderer** component to your cameras. This renders the amount of light that falls on each pixel of the screen. Secondly, you need to change your sprites to use the **SFSoftShadow** shader or one of **SF\*** materials. This will apply the light generated by the **SFRenderer** to your sprites. Third, you need to add lights to your scene using **SFLight**. Lastly, for objects that you want to cast a shadow, you need to add a **SFPolygon** component to let it know where it should cast the shadows.

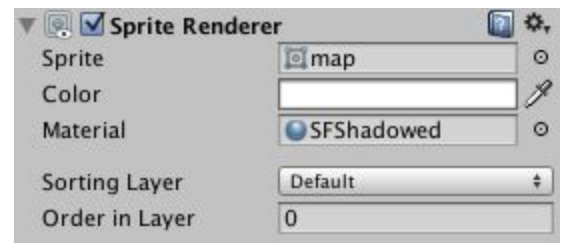
1. Add a **SFRenderer** component to your cameras:

- Go *Component > SFShadow > SFRenderer*
- It's a good idea to set the ambient light property of the **SFRenderer** to a bright color now. This makes it easier to debug setting up sprite materials later.



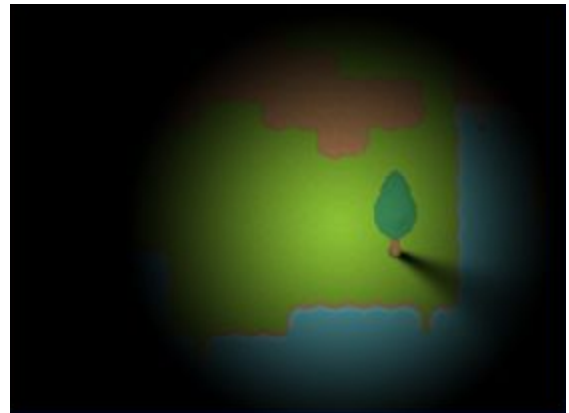
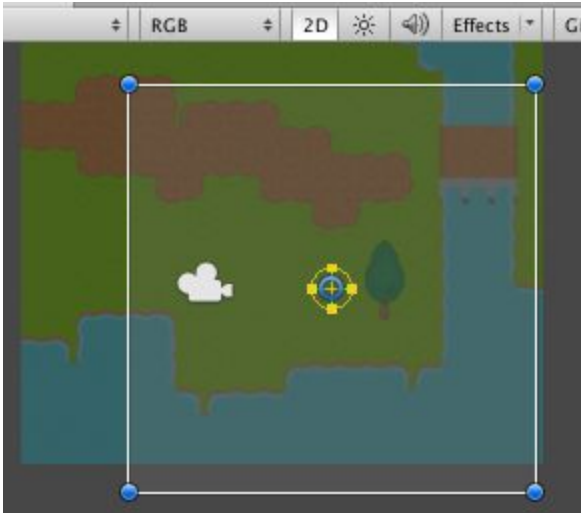
2. Use the **SFSoftShadow** shader or **SF\*** materials on your sprites:

- **SFUnshadowed** will cause your sprites to be lit by lights, but not have shadows cast on them. This is generally what you want for foreground sprites.
- **SFShadowed** will cause your sprites to be lit by lights, and also have shadows cast on them. This is generally what you want for background sprites.
- If you set a bright colored ambient light property of the **SFRenderer** in the last step, your sprites will be tinted by the ambient light color. This makes it easy to see that you didn't miss any.

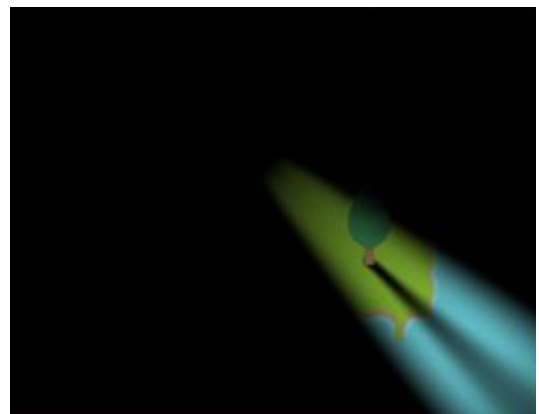
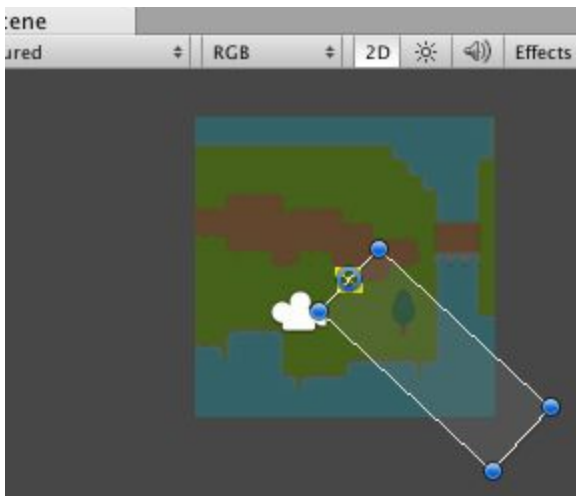


### 3. Add some **SFLights**:

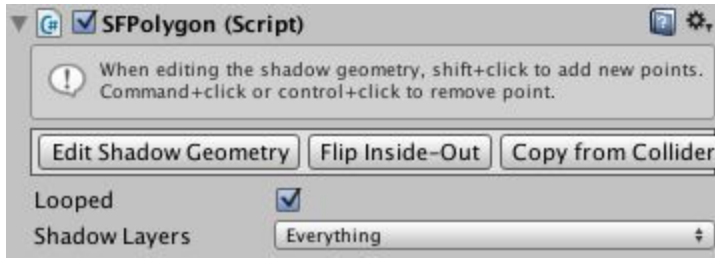
- First, pick a good ambient light color and set it on your SFRenderer. Black, or a very dark grey is best so that it doesn't hide the effect of the lights.
- You'll probably want to add new game objects to attach your lights to, so do that first.
- Go *Component > SFShadow > SFLight*
- Lights use a *RectTransform* to set up the light's properties. The light's cookie texture is drawn into the rect while the shadows are cast from the anchor point. This gives you the flexibility to set up different kinds of lights such as spotlights or point lights.
- A light with a radial gradient as its cookie texture, with its anchor point at its center will draw like this:



- A light with a spotlight cookie texture with the anchor point on the edge will draw like this:



4. Add **SFPolygon** components to objects you want to cast shadows:



- Objects that you want to cast shadows need to be outlined using SFPolygon components.
- *Go Component > SFShadow > SFPolygon*
- If your object already has a PolygonCollider2D component, the shadow geometry will be automatically copied from that. Otherwise click the “Edit Shadow Geometry” button to edit the polygon in the scene view.
- When in edit mode, shift clicking will add new points and control clicking will delete points.
- Shadow polygons are one sided. Click the “Flip Inside-Out” button to change between an inside loop or an outside loop on an object.
- Polygons don’t need to be looped. This is useful for large objects such as terrain that you want to split up.

5. (Optional) Add SFSample for upright objects in a 2.5D game.

- This is an example with line sampling turned on, and the sample point set to the bottom edge of the sprite:



- Normally a sprite samples the lightmap for each pixel. Using SFSample, you can make it sample the whole sprite from the same point. With line sampling enabled, the light for each column of pixels is sampled from a line that passes through the sample point.

# Scripts and Properties:

- **SFLight** - Basic light component. Hopefully self explanatory.
  - **radius** : *float* - The radius of the light source. Larger radii make for softer shadows, 0 makes hard shadows.
  - **color** : *Color* - The tint color of the light.
  - **cookieTexture** : *Texture* - An alpha texture that defines the shape of the light.
  - **shadowLayers** : *LayerMask* - Filter polygons to only render from the given layers.
  - **parallaxLight** : *bool* - Should the light cookie take the light's z-value into account?
- **SFPolygon** - Shadow casting component.
  - **pathCount** : *int* - The number of paths in the polygon.
  - **GetPath(int index)** - Get a path.
  - **SetPath(int index, Vector2[] pat)** - Set a path.
  - **looped** : *bool* - Should it draw a shadow edge between the first and last points in **verts**?
  - **shadowLayers** : *LayerMask* - The layers that a polygon is in.
  - **CopyFromCollider(PolygonCollider2D pc)** - Copy the paths from a collider. (Note: Does not handle paths for holes automatically)
  - ~~**verts** : *Vector2[]* - The list of vertexes for the polygon. (deprecated)~~
- **SFRenderer** - Camera component that renders the lighting effect.
  - **linearLightBlending** : *bool* - Nonlinear blending prevents oversaturation, but can cause draw order artifacts.
  - **shadows** : *bool* - Should shadows be rendered?
  - **ambientLight** : *Color* - The global ambient light.
  - **exposure** : *float* - A multiplier applied to all lights. Can cause oversaturation. (formerly global illumination scale)
  - **minLightPenetration** : *float* - Minimum allowed light penetration. Overrides the per polygon setting if greater.
  - **shadowCompensation** : *float* - Over-darken shadows to blend out seams on some devices. Recommended value is slightly more than 1.0.
  - **lightMapScale** : *float* - The resolution of the lightmap texture compared to the screen. Recommended values are between 8 - 32.
  - **shadowMapScale** : *float* - The resolution of the lightmap texture compared to the screen. Recommended values are between 2 - 4.
  - **fogColor** : *Color* - The color of the fog layer.
  - **scatterColor** : *Color* - The color that the fog will glow when lit.
  - **softHardMix** : *float* - What percentage of unshadowed/shadowed light should apply to the fog.
  - **extents** : *Rect* - Clip space rect that lights should be rendered for. Useful to extend shadow rendering offscreen to use with SFSample.
- **SFSample** - Script that controls how light sampling is performed. Useful for 2.5D games (isometric, etc)
  - **samplePosition** : *Vector2* - Position in local coordinates to sample the lightmap from. The light value at this position is used for the whole object.
  - **lineSample** : *bool* - When true, only the y-value of the sample point is used. Useful for things like shrubs/trees, and can be used for a simple self-shadowing effect.