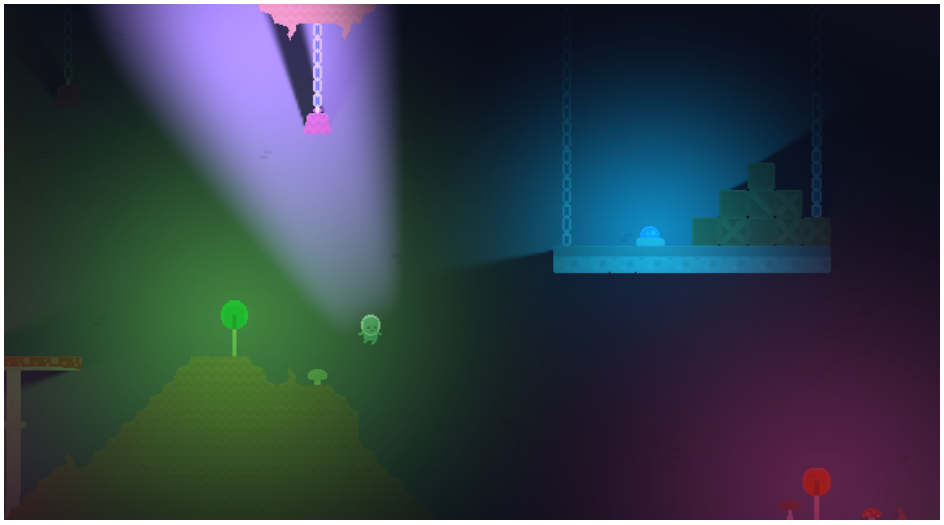
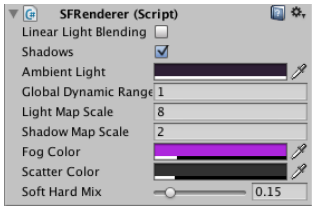
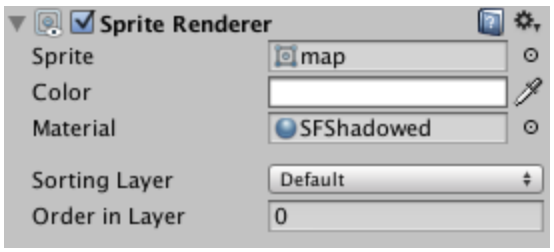
Super Fast Soft Shadows



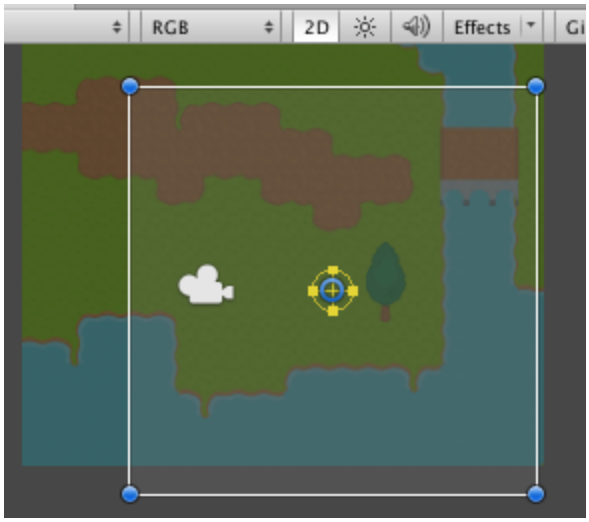
Adding support for Super Fast Soft Shadows to your project consists of four easy steps. First, you need to attach a SFRenderer component to your cameras. This renders the amount of light that falls on each pixel of the screen. Secondly, you need to change your sprites to use the SFSoftShadow shader or 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.

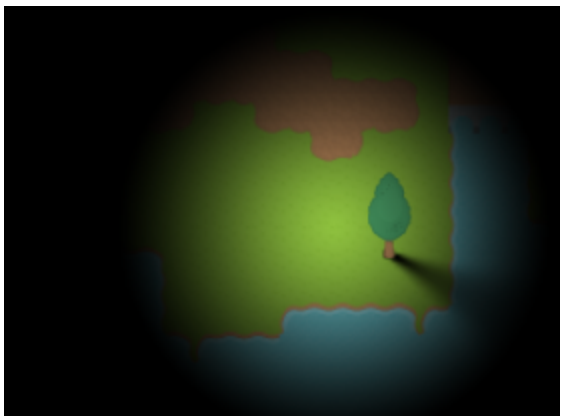


1. Add a **SFRenderer** component to your cameras:
    - Go *Component* > *SFShadow* > *SFRenderer*
    - It's a good idea to set the ambient light property of the SFRenderer to a bright color now. This this makes it easier to debug setting up sprite materials later.
2. Use the **SFSoftShadow** shader 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.

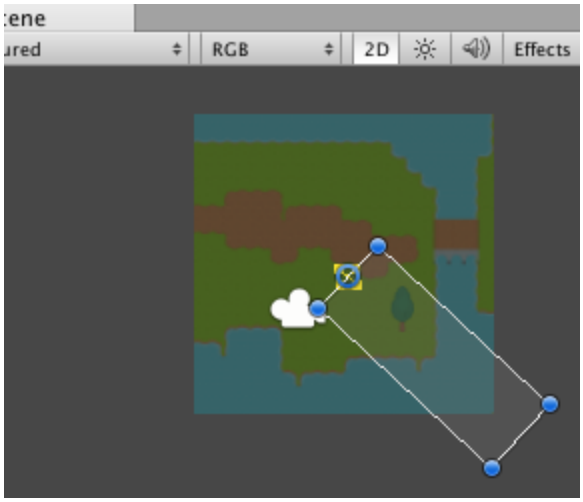


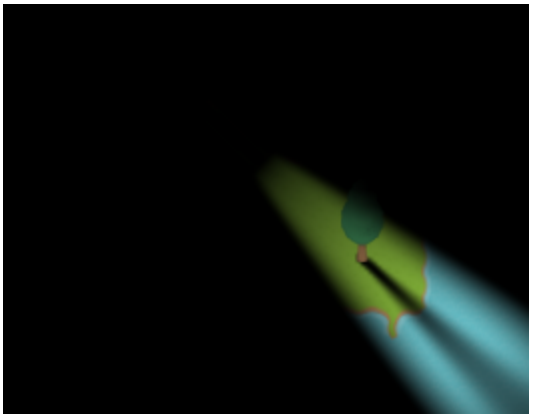
3. Add some **SFLights**:
    - First, pick a good ambient light color and set it on your SFRenderer. Black, or a very dark grey is best so that it doesn't hide the effect of the lights.
    - You'll probably want to add new game objects to attach your lights to, so do that first.
    - Go *Component* > *SFShadow* > *SFLight*
    - Lights use a 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:



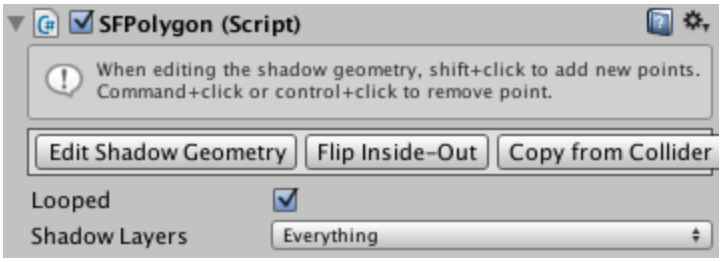


    - A light with a spotlight cookie texture with the anchor point on the edge will draw like this:





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.

Scripts and Properties:

- **SFLight**
  - **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.
- **SFPolygon**
  - **verts** : *Vector2[]* - The list of vertexes for the polygon.
  - **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.
- **SFRenderer**
  - **linearLightBlending** : *bool* - Nonlinear blending prevents oversaturation, but can cause draw order artifacts.
  - **shadows** : *bool* - Should shadows be rendered?
  - **ambientLight** : *Color* - The global ambient light.
  - **gloabalDynamicRange** : *float* - A multiplier applied to all lights. Can cause oversaturation.
  - **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.

Performance Tips:

- Soft shadow rendering performance is mostly affected by the number of lights onscreen at once, and how many pixels on the screen they cover. You want to make sure that on average, each pixel on the screen is affected by only a few lights.
- If you can’t reduce the average number of lights per pixel, then use the downsampling parameters on SFRenderer to draw fewer pixels.
- SFPolygons are batched on the CPU. This means that for very large and detailed polygons, such as a level outline, it’s better to break them into smaller chunks. That way the CPU doesn’t have to spend time batching parts of the polygon that can’t be seen. The cost is per-vertex, so only worry about large polygons if they have a lot of vertexes.
- Try disabling shadows on very small lights. This will save you a draw call and some CPU time for culling shadow casters.