Unity3D Tutorial – Materials and Particle Effects

0. About This Tutorial

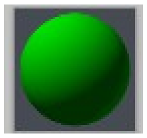
1. This is the second tutorial in a series of Unity3D tutorials for. To do this tutorial, you should be comfortable with basics of Unity3D user interface, have internet access, and have some experience drawing simple shapes in a drawing computer program.
2. This tutorial teaches basics of how to use **Materials** and **Textures**, but is primarily focused on using the Unity3D **Shuriken** particle system engine.

1. Starting Point

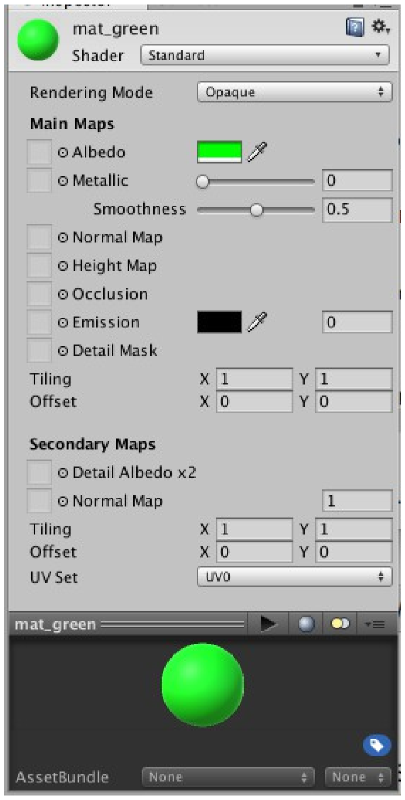
1. You can chose to do this tutorial in an existing project, or in a new project. If doing this in an existing project, skip this section, otherwise, within Unity3d, press File, and select New Project... .
2. Browse... to the Desktop, and create a New folder (**Right-Mouse Button** in the *File Chooser* and select New ▸ Folder, or just press **Ctrl + Shift +N** or **⌘+Shift+N**)
3. Name the folder unity3d material particles demo, then Select Folder. Press Create to create the project.

2. Create a Simple Material

1. Before creating a material, find an object in the scene to apply the material to! If you don't have one, create one in the Hierarchy, with Create, **3D Object**, Cube.



2. To create the material, in the Project, select Create and Material.
3. Name the material something like "mat_green".
4. Notice material options in the Inspector. The options belong to the material's **Shader**.
5. Click the color picker ( ) to the right of **Albedo** to change colors.
   - ***Albedo*** is a scientific word that means "light bouncing off of something"
6. Click inside the ▼ Colors *gradient* to change the color. To change the square: Click the rainbow on the right, drag the ▼ Sliders, or type a raw RGB value in the text boxes.
7. Close the window after picking a color.
8. *Drag-and-drop* ( ) the material onto a **GameObject** in the Hierarchy, or even in the Scene.
9. Create more colors by creating more Materials!



mat_green
Shader: Standard

Rendering Mode: Opaque

Main Maps
- Albedo
- Metallic: 0
- Smoothness: 0.5
- Normal Map
- Height Map
- Occlusion
- Emission: 0
- Detail Mask

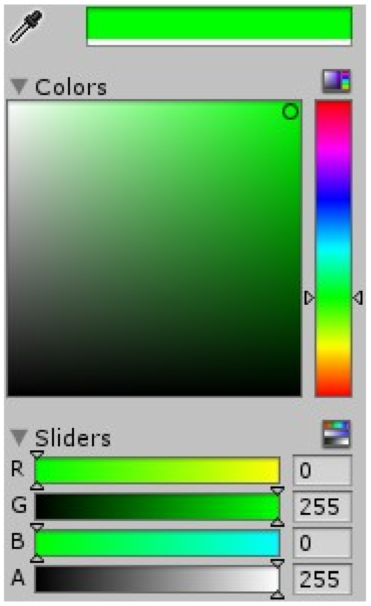
Tiling X 1 Y 1
Offset X 0 Y 0

Secondary Maps
- Detail Albedo x2
- Normal Map: 1

Tiling X 1 Y 1
Offset X 0 Y 0
UV Set: UV0

mat_green

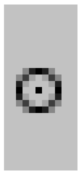
AssetBundle: None, None



▼ Colors

▼ Sliders
R: 0
G: 255
B: 0
A: 255

3. Texture and Shader Basics



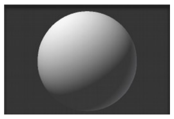
1 A **Material** can put a picture on a **GameObject** if the **Material** has a **Texture**. Click the ⊙ *tiny-circle-with-the-dot-in-it* next to **Albedo**, and select a texture. Every Unity3D project should have at least 1 texture (the default particle texture).

- The ***tiny-circle-with-the-dot-in-it*** is Unity's "select another resource" icon.

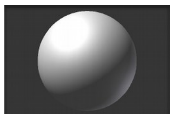
2 You may notice that setting a **Material**'s texture changes all **GameObject**s using that material.

3 A **Shader** changes how objects *light* themselves. Click the Shader [Standard ▾] *drop-down box* to see shader options. Don't worry about them all, just know they are there. Shaders are a big topic!

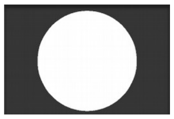
- Game Developers should know something about shaders:



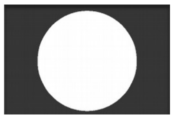
Diffuse – Also called *Flat Shading*. This is a basic idea that applies to many different kinds of shaders. Diffuse lighting makes something look correct in light, but not particularly special.



Specular – *Specular* is the term for the white circle of shininess on a reflective material. Objects with specular look more wet than objects with only **Diffuse** lighting. The smaller that white reflection, the shinier and wetter something looks.

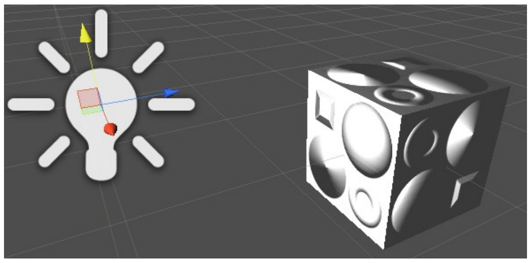


Unlit – An Unlit shader tells the 3D graphics system not to apply lighting calculations to this object. Unlit objects will look very artificial, and stand out in a game that is using other shaders.



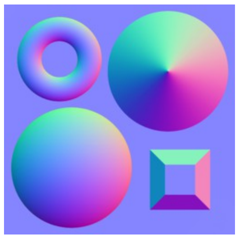
Particles/Additive – Many *particle effects* use this shader. It acts intelligently with transparency, and generates good-looking color blending, without costing the computer much processing power.

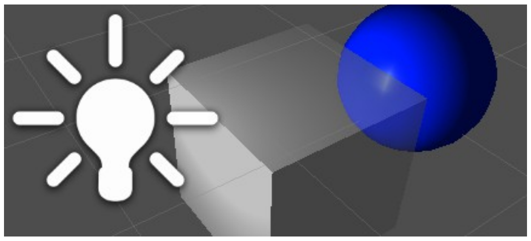
Sprites – These shaders intelligently handle transparency. Use these if you want to display images-with-transparency onto objects, especially as pieces of user interface that interact within a game area.



⊙ Normal Map The standard shader allows **Normal Map**s or *Bump Maps*, to show detail on a surface when lit. **Normal Map**s are an important part of very high-quality 3D art development for games.

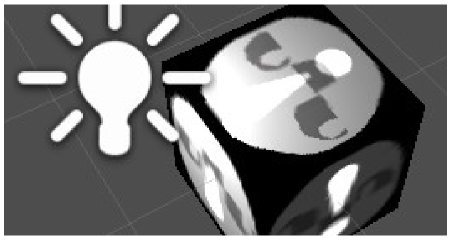
← White cube and lighting, using Normal Map →
http://codegiraffe.com/utut/normalmap.png →





Rendering Mode [Transparent ▾] The standard shader in Unity has a **Transparent Rendering Mode**, which allows textures to be partially see-through. The *Alpha Transparency* can be set in the **Material**'s color (the alpha transparency bar at the bottom of the color picker).

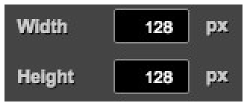
← Partially-transparent white cube in front of a specular blue sphere



Emission – The standard shader can apply a light, or a light stencil effect, directly onto textures. These light effects can help accent details, and make a texture look lit regardless of actual lighting.

4. Creating Simple Textures with PiskelApp for Unity3D

1. Unity3D can use almost any image type as a texture. Lets make a very simple image using PiskelApp, a free animated-sprite-editor on the internet, found at http://piskelapp.com/p/create .

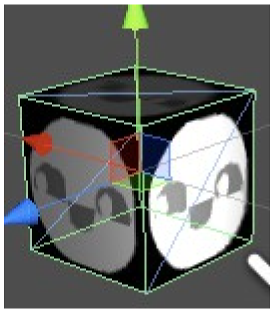


Width 128 px
Height 128 px

2. Press (on the right) to set the size of the starting image to 128 x 128. Notably, 128 is a *power of 2*, which is easier for a computer to use than any other kind of number (some graphics hardware *requires* images to be sized by powers of 2).
3. Colors are changed with the rectangles at the lower-left.
4. Use the (P)encil tool () and paint (B)ucket () to draw something with your mouse. I drew a silly face, and left the edges transparent.

Sample Placeholder Art

5. Name the graphic with Save (), and *download* it as a **.png** with Export ().
6. Don't spend time to make good art right now, *make fast art*! Game Developers call this *Placeholder Art*, it is meant to be replaced later, by an artist taking their time.
7. To import the image into Unity3D, save your image in the **Assets** folder of your Unity3D project. Or, you can also simply *drag-and-drop* the image into the **Assets** folder of your project (either in Unity3D, or in the file system). To see the **Assets** folder in the file system, press the **Right-Mouse Button** on **Assets**, then Show in Explorer or Reveal in Finder. or, use Assets, Import New Asset..., and then find the image using the *file chooser*.



8. You can bring any image into Unity3D, by following the previous step.
9. Image assets can be put onto objects by *drag-and-drop*ping them. Doing that will automatically create a **Material** in a Materials sub-folder, which is created in the same folder as the image.

Don't Forget to Save Your Scene!

5. Simple Particle Effects

1 Create a new **Particle System** in the Hierarchy by clicking Create ▾ and Particle System.

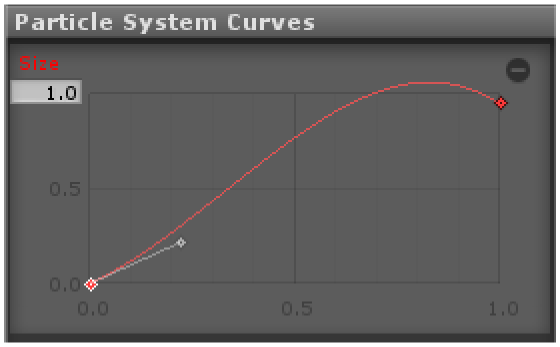
- The **Particle System Component** describes how the particle effect works, and can be changed by editing its values directly.
- Hover your mouse over the text label next to each value for a description of what the value is for.
- The Particle System has many sub-components (like **Emission**, **Shape**, **Size over Lifetime**). Click on the sub-components of the **Particle System** to expand them. Click the white circle to toggle them on and off.
- You can rename the **Particle System** to anything. For this demonstration, name it "Absorb from Environment".

2 Copy the **Emission** and **Shape** sub-components to match the example on the right. →

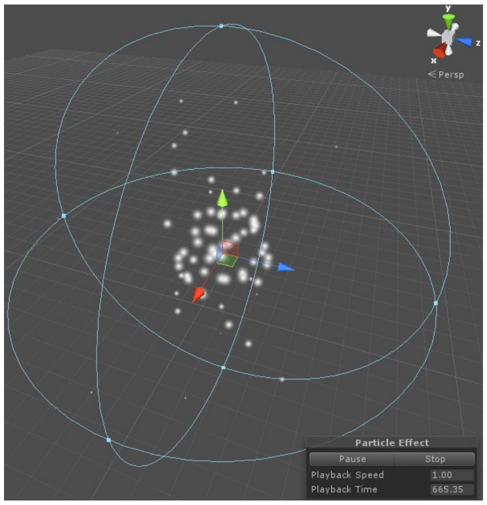
3 Enable and expand **Size over Lifetime**, and click in the gray rectangle. Clicking the rectangle activates the **Curve Editor**.

4 You may need to expand the **Particle System Curves** editor by dragging it taller: hover your ⇕ mouse over the **Particle System Curves** title, near the bottom of the inspector, and drag it upward.

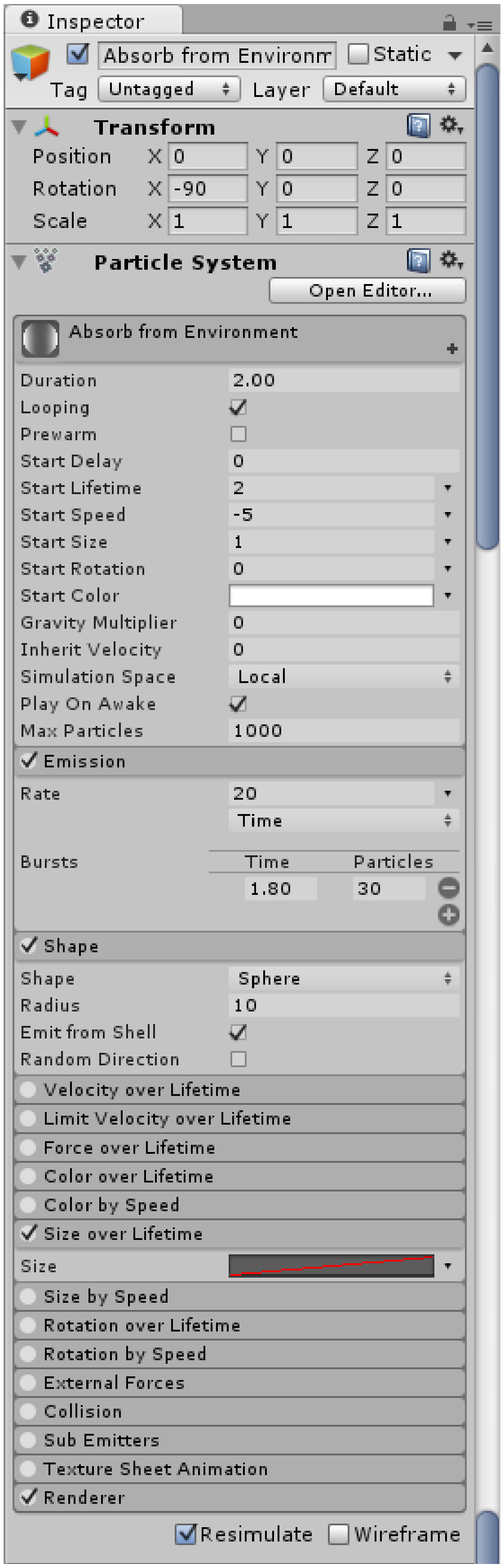
5 The **curve editor** modifies a *spline*, a special kind of curved line. Drag spline points, or adjust the slope with a handle that appears after selecting a point.



6 Notice that the shape of the spline determines the size of particles at different parts of the particle's life.



- With these values (the Inspector sample to the right), this **Particle System** creates a sphere area that pulls white energy of some kind towards a center point.
- Don't forget to save your work!
- Experiment with different numbers to see what happens to the **Particle System**.



Inspector

✓ Absorb from Environm | Static
Tag Untagged | Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	-90	0	0
Scale	1	1	1

Particle System — Open Editor...

Absorb from Environment

Duration	2.00
Looping	✓
Prewarm	
Start Delay	0
Start Lifetime	2
Start Speed	-5
Start Size	1
Start Rotation	0
Start Color	⬜
Gravity Multiplier	0
Inherit Velocity	0
Simulation Space	Local
Play On Awake	✓
Max Particles	1000

✓ Emission

Rate	20	
	Time	
Bursts	Time	Particles
	1.80	30

✓ Shape

Shape	Sphere
Radius	10
Emit from Shell	✓
Random Direction	

Velocity over Lifetime
Limit Velocity over Lifetime
Force over Lifetime
Color over Lifetime
Color by Speed
✓ Size over Lifetime
Size
Size by Speed
Rotation over Lifetime
Rotation by Speed
External Forces
Collision
Sub Emitters
Texture Sheet Animation
✓ Renderer

✓ Resimulate | Wireframe

6. Particles Using Materials

1 Get the image Star02.png from https://goo.gl/LCw5Oo

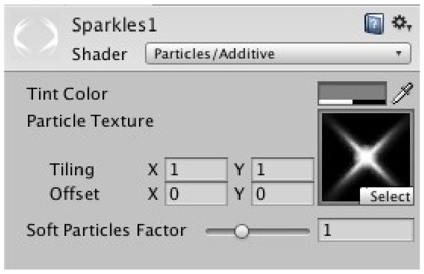
2 Add this image to your Unity **Assets**.

3 Create a new **Material**, and set it's shader to **Particles/Additive**

- **Particles/Additive** tells a shader to treat black as transparent, and white as maximum color, with shades of gray being partially transparent. When combined with different colors, and many particles, this allows very pretty color-blending effects for a material!

4 Name the new material **Sparkles1**

5 Use Select to choose Start02.png as the texture for the material, or *drag-and-drop* Start02.png into the Select box.



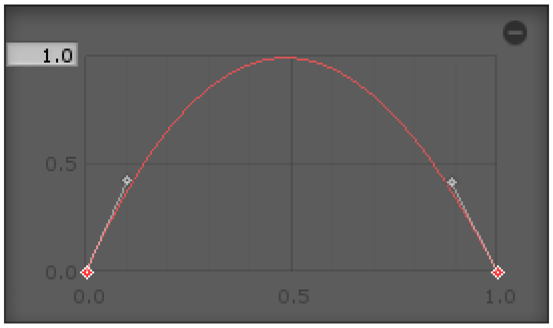
Sparkles1
Shader: Particles/Additive

Tint Color	
Particle Texture	Select
Tiling	X 1 Y 1
Offset	X 0 Y 0
Soft Particles Factor	1

6 Create a new **Particle System** in the Hierarchy by clicking Create and Particle System and name it twinkle.

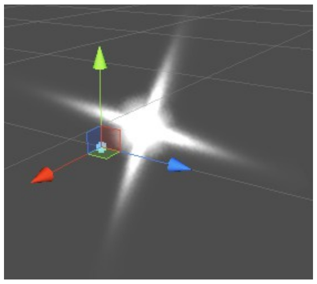
- Please keep this particle effect simple! It is designed to be a subtle *accent*. Small effects like this can give nice hints to players.

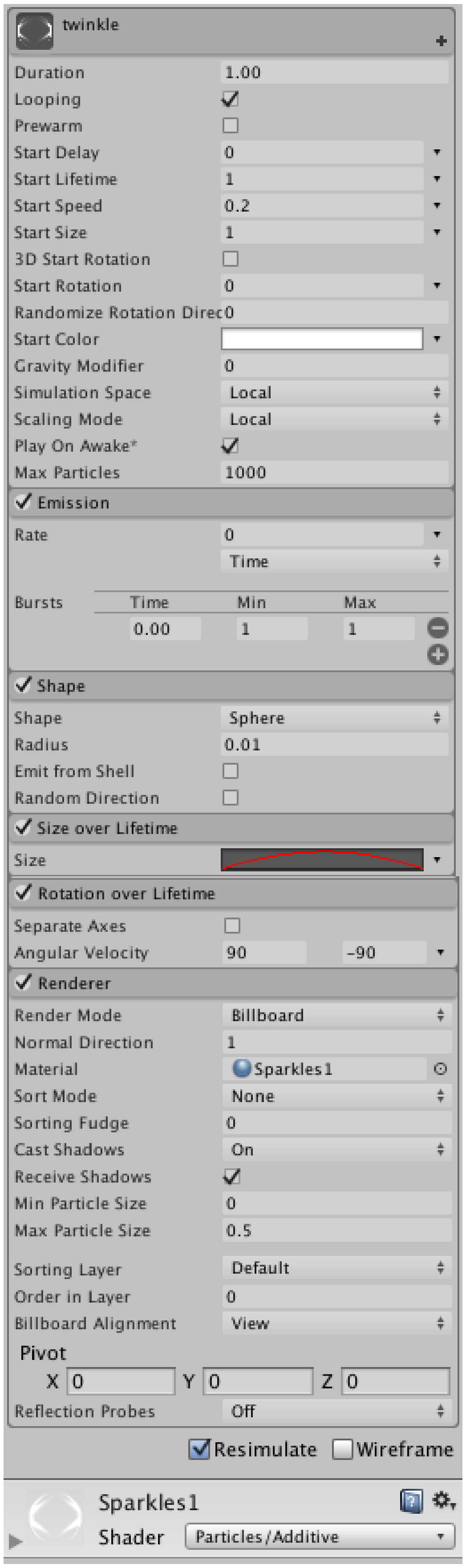
7 **Size over Lifetime** should be enabled, and should use a parabola-like curve (here →).



8 **Renderer** should use the **Material**: **Sparkles1** (press ⊙ to find it).

9 **Rotation over Lifetime** should be enabled. Change the **Angular Velocity** value to accepting two values by pressing *the-tiny-down-arrow* ▾ and selecting Random Between Two Constants. Use -90 and 90.





twinkle

Duration	1.00
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	1
Start Speed	0.2
Start Size	1
3D Start Rotation	☐
Start Rotation	0
Randomize Rotation Direc	0
Start Color	
Gravity Modifier	0
Simulation Space	Local
Scaling Mode	Local
Play On Awake*	✓
Max Particles	1000

✓ Emission

Rate	0		
	Time		
Bursts	Time	Min	Max
	0.00	1	1

✓ Shape

Shape	Sphere
Radius	0.01
Emit from Shell	☐
Random Direction	☐

✓ Size over Lifetime

Size	

✓ Rotation over Lifetime

Separate Axes	☐	
Angular Velocity	90	-90

✓ Renderer

Render Mode	Billboard
Normal Direction	1
Material	Sparkles1
Sort Mode	None
Sorting Fudge	0
Cast Shadows	On
Receive Shadows	✓
Min Particle Size	0
Max Particle Size	0.5
Sorting Layer	Default
Order in Layer	0
Billboard Alignment	View
Pivot	X 0 Y 0 Z 0
Reflection Probes	Off

✓ Resimulate ☐ Wireframe

Sparkles1
Shader: Particles/Additive

7. Particles Using Sprite Materials

1 Create an image to act as a sprite, using your favorite image editor.

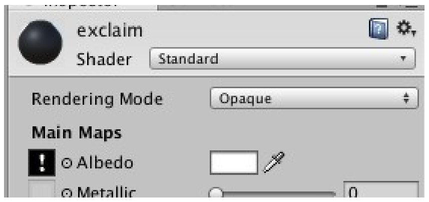
- This particle effect is designed to be an iconic alert! A simple, iconic image would work well. For example:

  https://goo.gl/iqyuL0

2 After creating the iconic image, **Import** it into Unity3D.

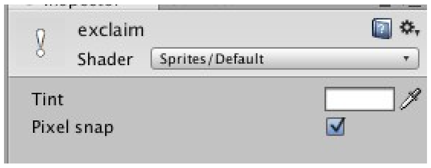
3 Create a new material called Exclaim.

4 Change the **Albedo Main Map** to the image for this particle effect by setting it with the ⊙ *tiny-circle-with-the-dot-in-it*.



exclaim
Shader: Standard
Rendering Mode: Opaque
Main Maps
⊙ Albedo
⊙ Metallic 0

5 After the image is set, change the **Shader** from **Standard** to **Sprites/Default**.



exclaim
Shader: Sprites/Default
Tint
Pixel snap ✓

- The image must be set first, in the **Standard** shader, because some versions of Unity don't support changing the **Albedo** texture in **Sprites** shaders.

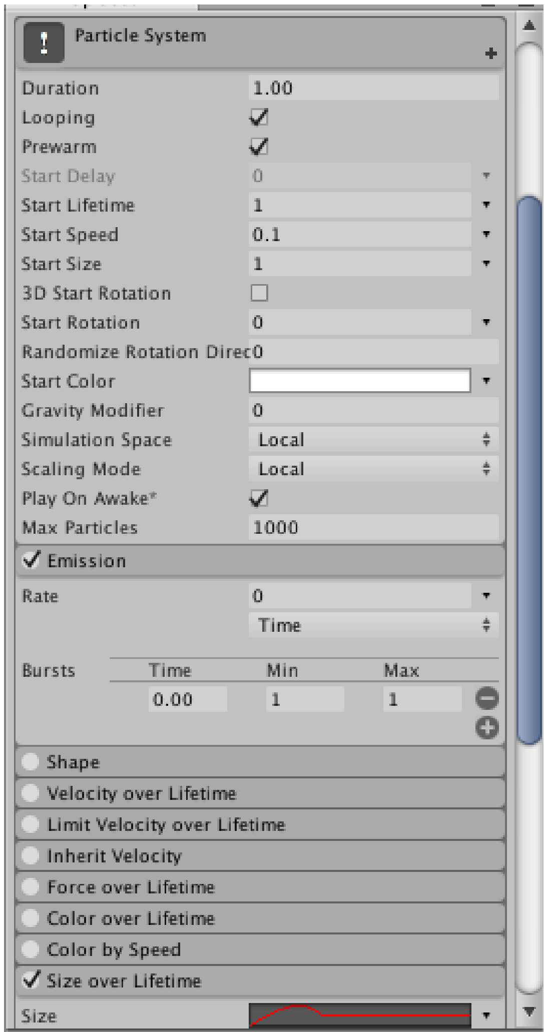
6 Make a new **Particle System**, using values to the right →, and name it exclaim.

7 *Un-check the* **Shape**.

- Particle Systems without a shape will simply shoot particles in the **Transform**'s forward direction. This can also be very helpful for projectile effects!

8 **Size over Lifetime** should be enabled, and should use a parabola-like curve with a flat finish (here →).

9 **Renderer** should use the **Material**: Exclaim (press ⊙ to find it, or drag the material into the **Renderer**'s material).



Particle System

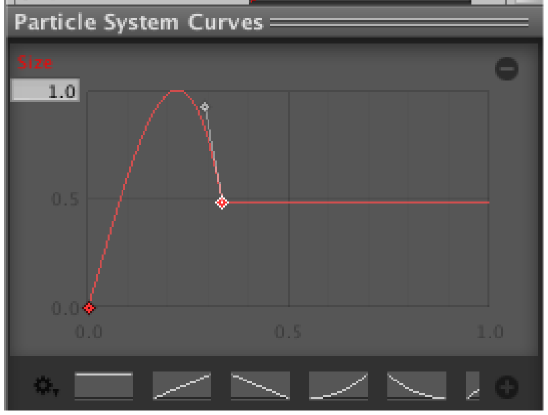
Duration	1.00
Looping	✓
Prewarm	✓
Start Delay	0
Start Lifetime	1
Start Speed	0.1
Start Size	1
3D Start Rotation	☐
Start Rotation	0
Randomize Rotation Direc	0
Start Color	
Gravity Modifier	0
Simulation Space	Local
Scaling Mode	Local
Play On Awake*	✓
Max Particles	1000

✓ Emission

Rate 0, Time

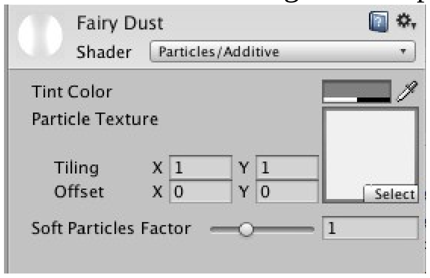
Bursts	Time	Min	Max
	0.00	1	1

Shape
Velocity over Lifetime
Limit Velocity over Lifetime
Inherit Velocity
Force over Lifetime
Color over Lifetime
Color by Speed
✓ Size over Lifetime
Size

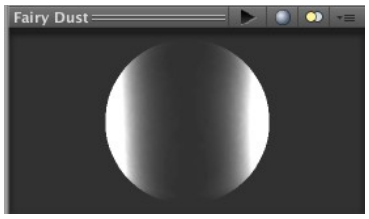


8. Particles With Blending Colors

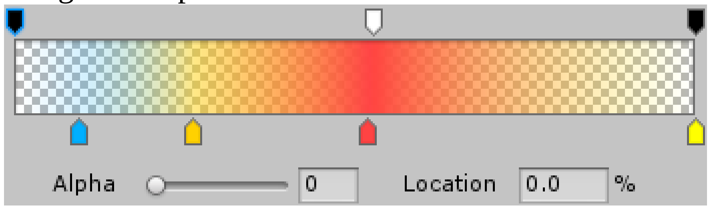
1. This particle works well with a partially transparent blob, like fairydust.tif, which can be found here: https://goo.gl/owdkHy
2. Create a material using the transparent blob-like image:



Fairy Dust
Shader: Particles/Additive
Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Select
Soft Particles Factor 1

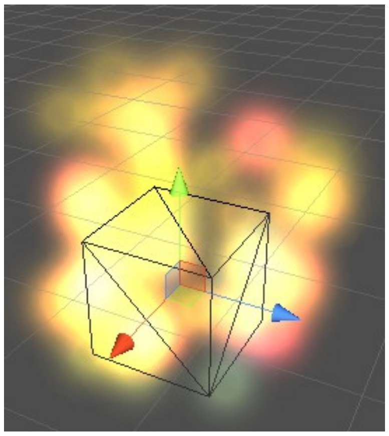


3. Make a new **Particle System**, using the values to the right, and name it "faerie fire".
4. **Color over Lifetime** should be enabled. There should be 4 color *markers* that can be selected, dragged, and modified using a color picker.

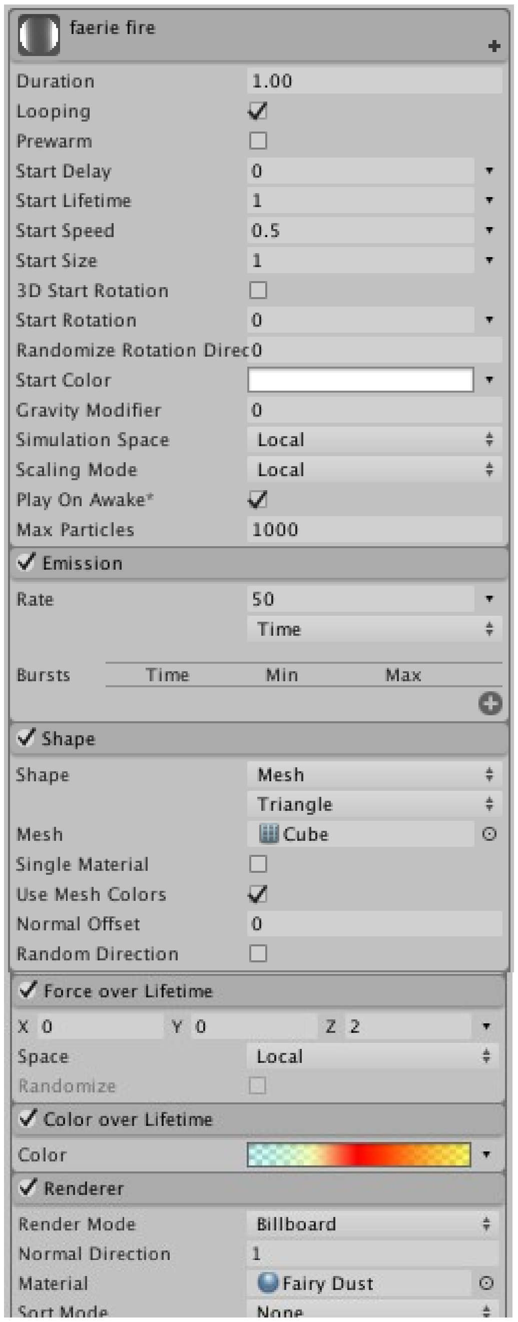


Alpha 0 Location 0.0 %

5. Create additional *markers* as pictured above by simply clicking in an empty area above or below the color gradient. Remove *markers* by pulling them off of the gradient editor.
6. Select the markers at the top to change the **Alpha Transparency**. The markers at the top-left and top-right should use an Alpha of 0, so that the particles fade in and out. Create a new marker at the top, and give it full Alpha.



- Note: Be careful with the **Material Editor** that appears below the **Particle Editor** in the **Inspector**! It provides *global access* to the **Material**! That means modifying this material will modify this particle effect, *and* every other object using this material!

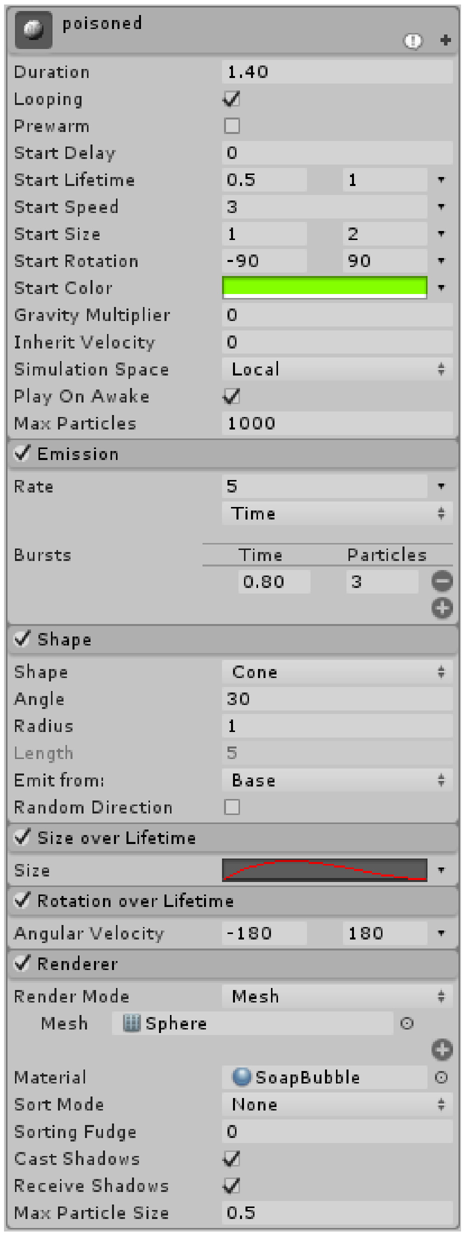


faerie fire

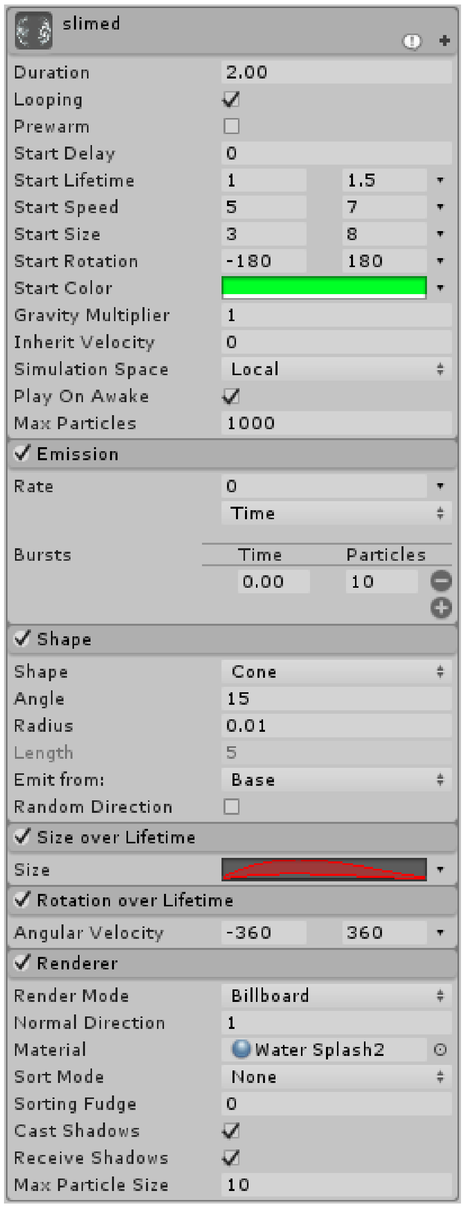
Duration	1.00
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	1
Start Speed	0.5
Start Size	1
3D Start Rotation	☐
Start Rotation	0
Randomize Rotation Direc	0
Start Color	⬜
Gravity Modifier	0
Simulation Space	Local
Scaling Mode	Local
Play On Awake*	✓
Max Particles	1000
✓ Emission	
Rate	50
	Time
Bursts	Time Min Max
✓ Shape	
Shape	Mesh
	Triangle
Mesh	Cube
Single Material	☐
Use Mesh Colors	✓
Normal Offset	0
Random Direction	☐
✓ Force over Lifetime	
X 0 Y 0 Z 2	
Space	Local
Randomize	☐
✓ Color over Lifetime	
Color	▨
✓ Renderer	
Render Mode	Billboard
Normal Direction	1
Material	Fairy Dust
Sort Mode	None

9. More Particle Effects

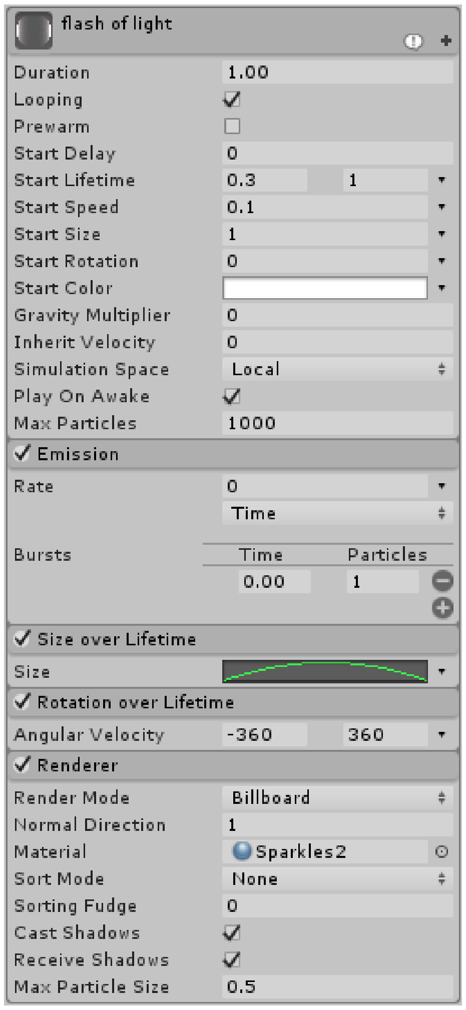
- Try making these **Particle Systems** for practice! *Note: these were created with Unity version 4!*



poisoned		
Duration	1.40	
Looping	✓	
Prewarm	☐	
Start Delay	0	
Start Lifetime	0.5	1
Start Speed	3	
Start Size	1	2
Start Rotation	-90	90
Start Color	🟩	
Gravity Multiplier	0	
Inherit Velocity	0	
Simulation Space	Local	
Play On Awake	✓	
Max Particles	1000	
✓ Emission		
Rate	5	
	Time	
Bursts	Time	Particles
	0.80	3
✓ Shape		
Shape	Cone	
Angle	30	
Radius	1	
Length	5	
Emit from:	Base	
Random Direction	☐	
✓ Size over Lifetime		
Size		
✓ Rotation over Lifetime		
Angular Velocity	-180	180
✓ Renderer		
Render Mode	Mesh	
Mesh	Sphere	
Material	SoapBubble	
Sort Mode	None	
Sorting Fudge	0	
Cast Shadows	✓	
Receive Shadows	✓	
Max Particle Size	0.5	

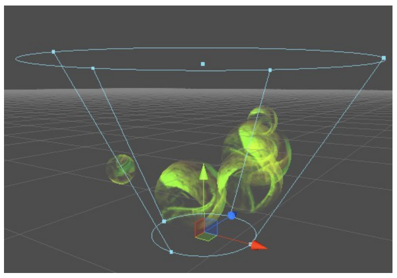


slimed		
Duration	2.00	
Looping	✓	
Prewarm	☐	
Start Delay	0	
Start Lifetime	1	1.5
Start Speed	5	7
Start Size	3	8
Start Rotation	-180	180
Start Color	🟩	
Gravity Multiplier	1	
Inherit Velocity	0	
Simulation Space	Local	
Play On Awake	✓	
Max Particles	1000	
✓ Emission		
Rate	0	
	Time	
Bursts	Time	Particles
	0.00	10
✓ Shape		
Shape	Cone	
Angle	15	
Radius	0.01	
Length	5	
Emit from:	Base	
Random Direction	☐	
✓ Size over Lifetime		
Size		
✓ Rotation over Lifetime		
Angular Velocity	-360	360
✓ Renderer		
Render Mode	Billboard	
Normal Direction	1	
Material	Water Splash2	
Sort Mode	None	
Sorting Fudge	0	
Cast Shadows	✓	
Receive Shadows	✓	
Max Particle Size	10	

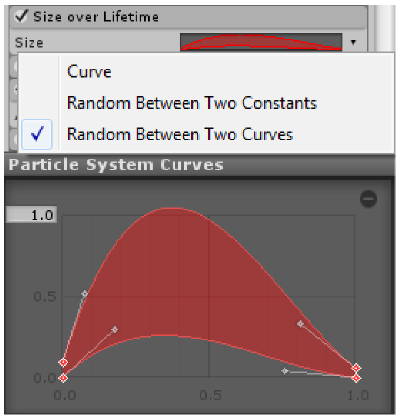


flash of light		
Duration	1.00	
Looping	✓	
Prewarm	☐	
Start Delay	0	
Start Lifetime	0.3	1
Start Speed	0.1	
Start Size	1	
Start Rotation	0	
Start Color	⬜	
Gravity Multiplier	0	
Inherit Velocity	0	
Simulation Space	Local	
Play On Awake	✓	
Max Particles	1000	
✓ Emission		
Rate	0	
	Time	
Bursts	Time	Particles
	0.00	1
✓ Size over Lifetime		
Size		
✓ Rotation over Lifetime		
Angular Velocity	-360	360
✓ Renderer		
Render Mode	Billboard	
Normal Direction	1	
Material	Sparkles2	
Sort Mode	None	
Sorting Fudge	0	
Cast Shadows	✓	
Receive Shadows	✓	
Max Particle Size	0.5	

The graphic for the SoapBubble material can be found here: https://goo.gl/9XZfs7

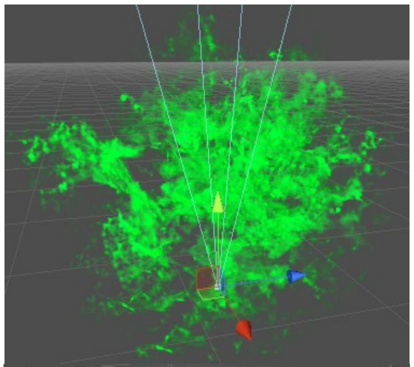


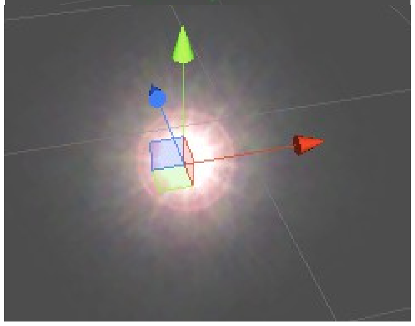
For the slimed particle effect, use 2 curves in **Size over Lifetime**:



✓ Size over Lifetime
Size

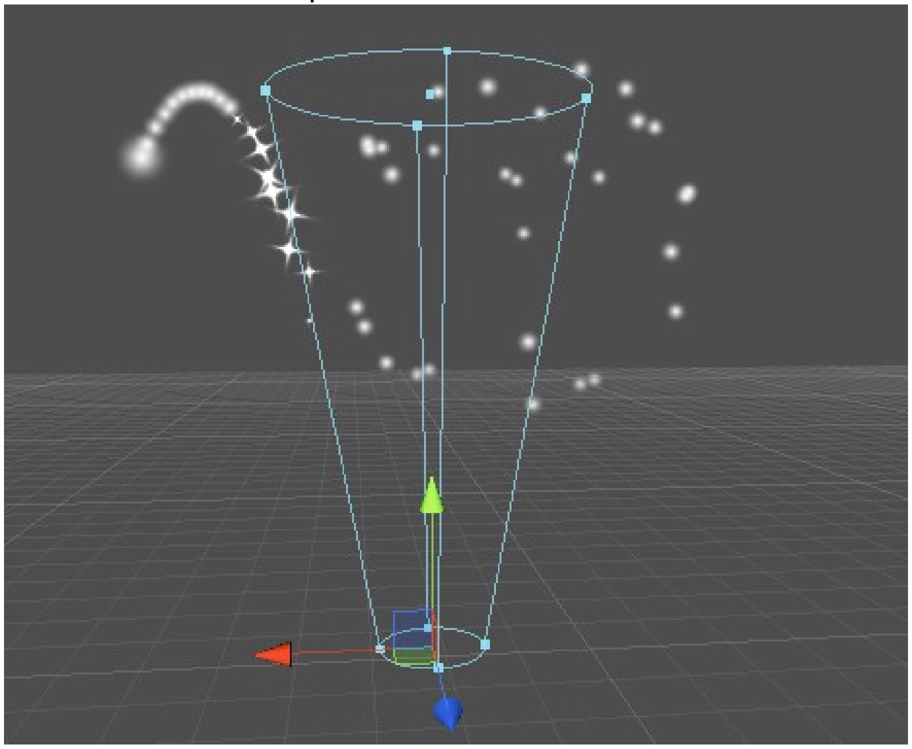
- Curve
- Random Between Two Constants
- ✓ Random Between Two Curves



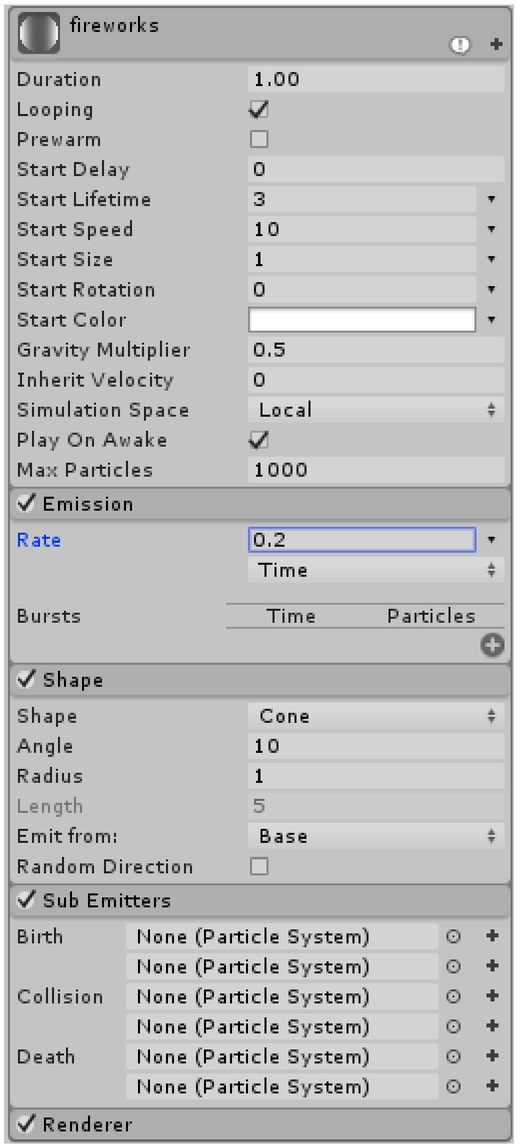


10. Particles With Sub Emitters

1. Start by creating the simple Particle System to the right →.
2. To the right of **Birth**, in the **Sub Emitters** sub-component, press the + button to add a new birth particle effect.
3. Notice that the fireworks **Particle System** now has a trail for each particle. Also notice in the Hierarchy that the fireworks **Particle System** now has SubEmitter as a child element. Rename that SubEmitter **Particle System** to fireworkTrail. Birth fireworkTrail (Particle Syst ⊙
4. Select fireworkTrail in the Hierarchy.
5. Using the ⊙ button to the right of **Death**, set the Sub Emitter of fireworkTrail to twinkle. Unity3D will ask if it should re-parent twinkle. Press No to keep things simple for now. Notice how the fireworkTrail twinkles away now. Death twinkle (Particle System) ⊙
6. Select fireworks again. Press the + button to add a new **Death** particle effect in **Sub Emitters**. Notice that the fireworks particles now explode. Also notice in the Hierarchy that the fireworks has another SubEmitter child element. Rename it to fireworkExplode.



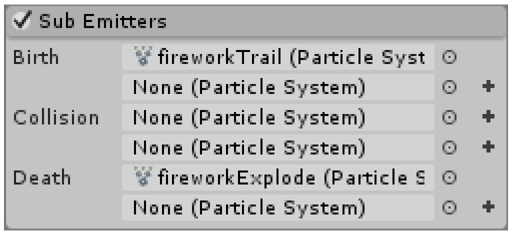
7. Try adding sub-emitters to fireworkExplode!



fireworks

Duration	1.00
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	3
Start Speed	10
Start Size	1
Start Rotation	0
Start Color	
Gravity Multiplier	0.5
Inherit Velocity	0
Simulation Space	Local
Play On Awake	✓
Max Particles	1000
✓ Emission	
Rate	0.2
	Time
Bursts	Time / Particles
✓ Shape	
Shape	Cone
Angle	10
Radius	1
Length	5
Emit from:	Base
Random Direction	☐
✓ Sub Emitters	
Birth	None (Particle System)
	None (Particle System)
Collision	None (Particle System)
	None (Particle System)
Death	None (Particle System)
	None (Particle System)
✓ Renderer	

The fireworks Particle System should have two **Sub Emitters**.



✓ Sub Emitters

Birth	fireworkTrail (Particle Syst
	None (Particle System)
Collision	None (Particle System)
	None (Particle System)
Death	fireworkExplode (Particle S
	None (Particle System)