Unity3D Tutorial – Materials and Particle Effects

0. About This Tutorial

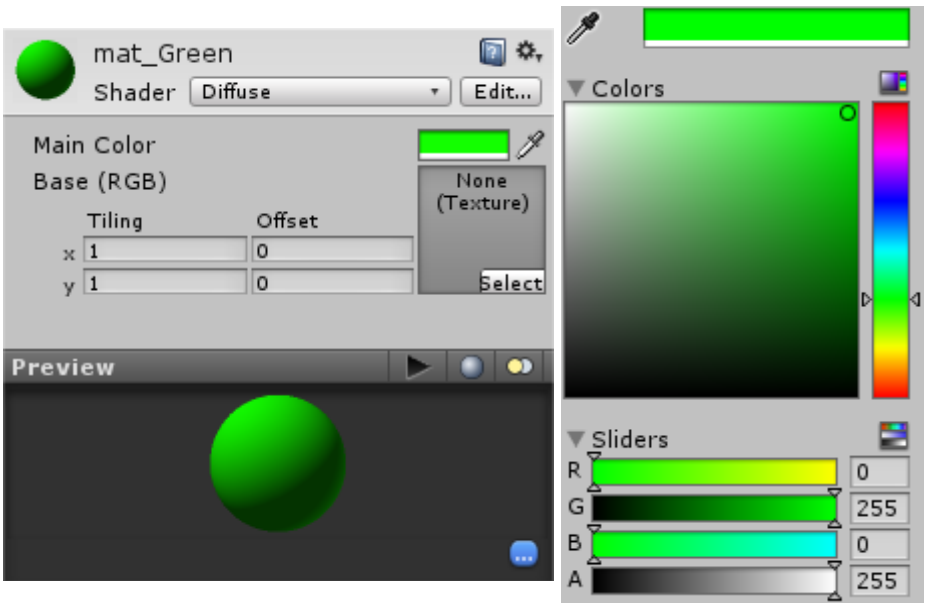
1. This is the second tutorial in a series of Unity3D tutorials for Windows. To do this tutorial, you should be comfortable with basics of Unity3D user interface, and Windows concepts.
2. This tutorial teaches basics of how to use **Materials** and **Textures**, but is primarily focused on using the 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**)
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 and 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.
5. *Click the material name in the Inspector to toggle options.* Doing this on accident can be very confusing!
6. Click the color picker ( ) to change colors.
7. Click inside the Colors *square* 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.
8. When done, click .

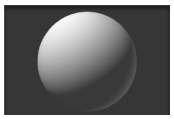


9. To apply this material to any object, simply drag-and-drop ( ) the material onto a **GameObject** in the Hierarchy, or even in the Scene.
10. Create more colors by creating more Materials!

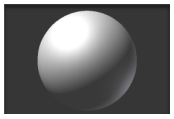
3. Texture and Shader Basics

1. A **Material** can put a picture on a **GameObject** if the **Material** has a **Texture**. Click the **Select** button in the gray rectangle that says "None (Texture)", and select a texture. Every Unity3D project should have at least 1 texture (the default particle texture).
2. Notice that setting a **Material**'s texture changes all **GameObject**s using that material.
3. A **Shader** changes the way that objects *light* themselves. Click the Shader Diffuse drop-down box to see available shader options. Don't worry about trying them all, just know they are there. Shaders are a big topic!

- Game Developers should know something about these basic shaders:



Diffuse – Also called *Flat Shading*. This is the basic shader, used by default. Using Diffuse lighting makes something look correct in light, but not particularly special.

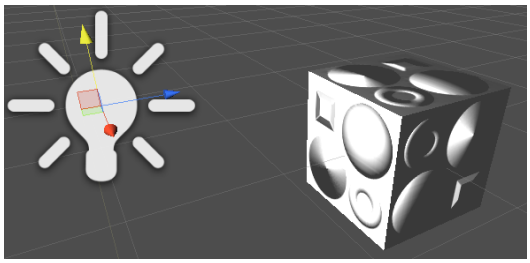


Specular – *Specular* is the term for the white speck of shininess on a reflective material. Objects with specular look more wet than objects with **Diffuse** lighting. The smaller that white reflection, the shinier and wetter something looks.

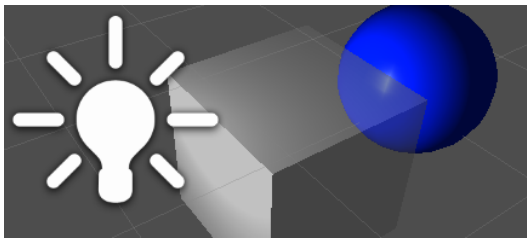
Unlit – The Unlit shader tells the 3D graphics system not to apply any lighting to this object. Unlit objects will look very artificial, and stand out in a game that is using other shaders.

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.

GUI/TextShader – Similar to **Particles/Additive**, it intelligently handles transparency, but it does not blend well. Use it if you want to display images-with-transparency onto objects, especially as pieces of user interface.

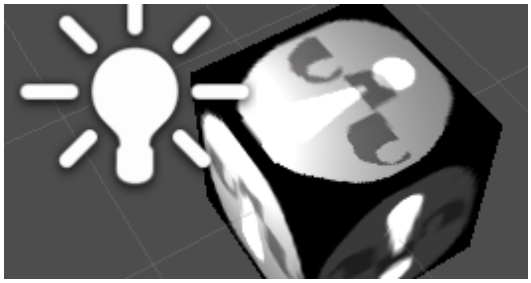


Bumped/Diffuse, **Bumped/Specular** – use **Normal Maps**, or *Bump Maps*, to show detail on a surface when lit. **Normal Maps** are an important part of very high-quality 3D art development for games.
← White cube and lighting, using Normal Map →
http://codegiraffe.com/utut/normalmap.png →



Transparent (Diffuse) – Transparent shaders allow an object 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



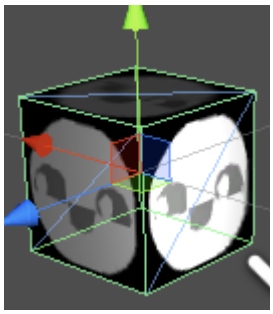
Self-Illumin – These shaders apply a light, or a light stencil effect, directly onto textures. These light effects can help accent details, and make a texture look lit regardless of actual lighting.

4. Creating Simple Textures for With MSPaint for Unity3D

1. Unity3D can use almost any image type as a texture. Lets make a very simple image using Windows' built in image editor **mspaint**, and use it as a custom texture,.
2. Start **mspaint** by selecting it from **All Programs** → **Accessories** → **Paint**, or by typing "mspaint" into the search are of the **Start Menu** or **Windows Dashboard**.
3. Press **Ctrl + E** to set the size of the starting image to Width: 128 Height: 128. Notably, 128 is a *power of 2*, which is easier for a computer to use than any other kind of number (some high-performance graphics hardware *requires* images to be sized by powers of 2).
4. Click the Pencil tool (), and use it to draw a simple face by dragging the mouse across the white area of the new image.
5. To help draw more precisely, zoom in with the **Magnifier** tool . With this tool, use the **Left-Mouse-Button** to zoom in, and **Right-Mouse-Button** to zoom out.
6. Fill areas using the fill tool () by clicking on an area of the image.
7. Use the color palette () on the right to change color used.
8. You can select and move or scale parts of an image using the **Selection** tool.
9. Don't spend time to make good art right now, *make fast art*! Game Developers call this ***Placeholder Art***, it's meant to be replaced later, by an artist taking their time.

Sample Placeholder Art

10. To import the image into Unity3D, save your image (**Ctrl + S**) 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 Windows file system). To see the **Assets** folder in *Windows Explorer*, press the **Right-Mouse Button** on Assets, then Show in Explorer. Or, you can use the Assets menu, Import New Asset..., and then find the image using the *file chooser*.
11. You can bring any image you want into Unity3D, by following the previous step.
12. 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.

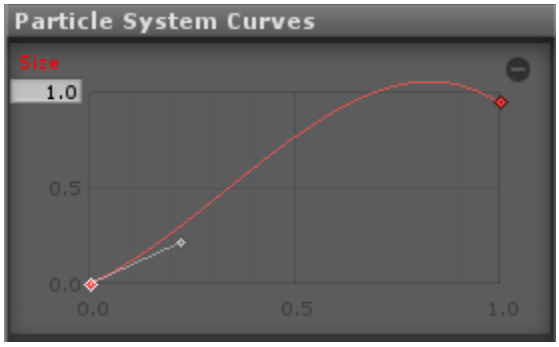
- The **Particle System Component** describes how the particle effect works, and can be changed by editing its values directly.
- Hover your mouse over the text label next to each value for a description of what the value is for.
- The Particle System has many sub-components (like **Emission**, **Shape**, **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. →

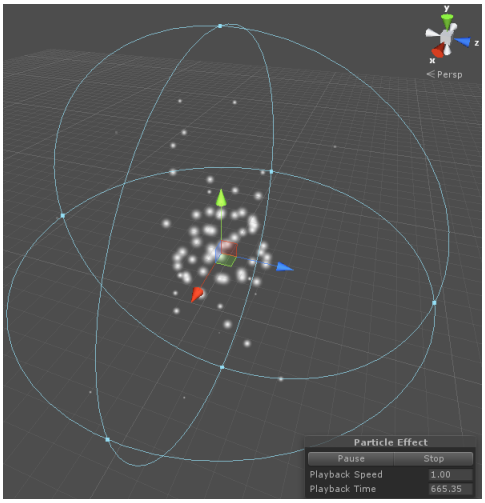
3 Enable and expand **Size over Lifetime**, and click in the gray rectangle. Clicking the rectangle activates the **Curve Editor**.

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.

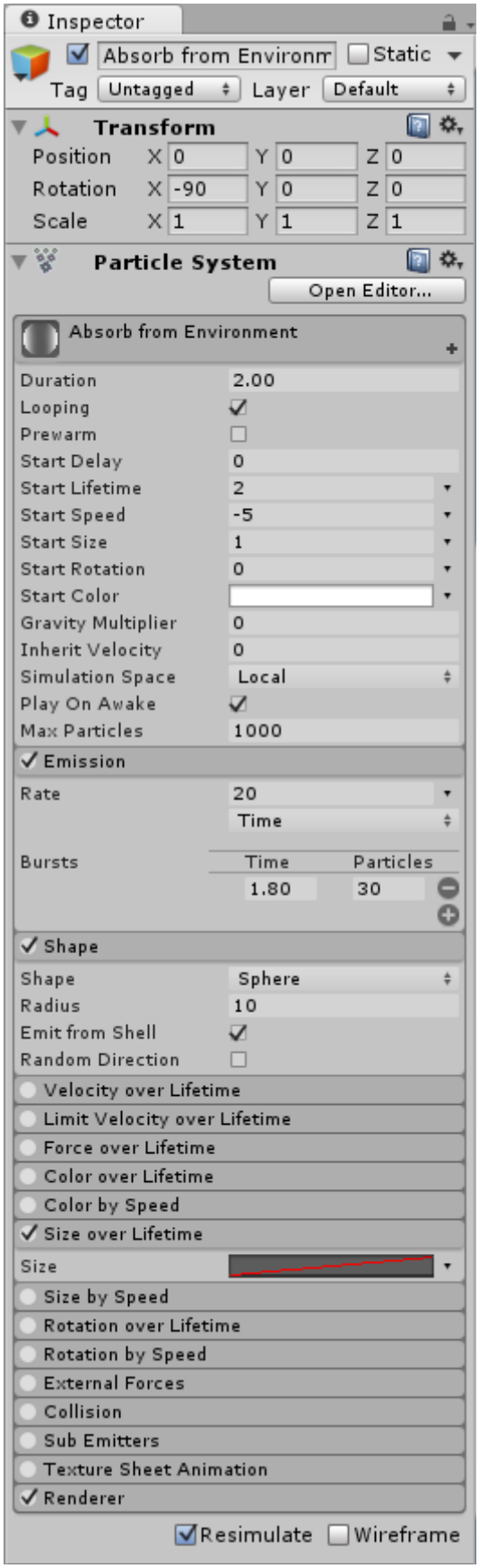
5 The **curve editor** modifies a *spline*, a special kind of curved line. Drag spline points, or adjust the slope with a handle that appears after selecting a point.



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**.



6. Particles Using Materials

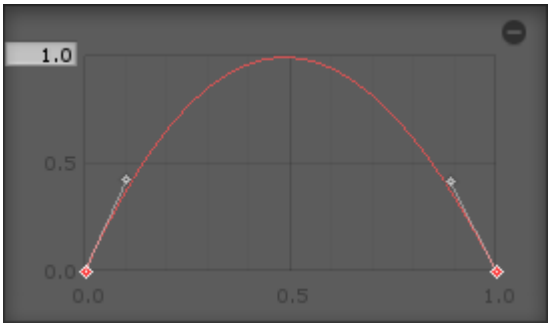
1. Import some useful **Materials** from the **Standard Assets** Library: select the Assets menu, then Import Package ▸ Character Controller, Particles.
2. To keep things simple, just press Import, and wait for it to load. Or to be more specific, press None, and then check ☑ the **Standard Assets/Particles/Sources/Materials** and **Standard Assets/Particles/Sources/Textures** folders.

- Note: Be careful about editing these imported **Materials**! If you want to modify these, it is safest to (1) make a copy of the Material, (2) rename the copy to be clearly different, and (3) adjust the *copy*!

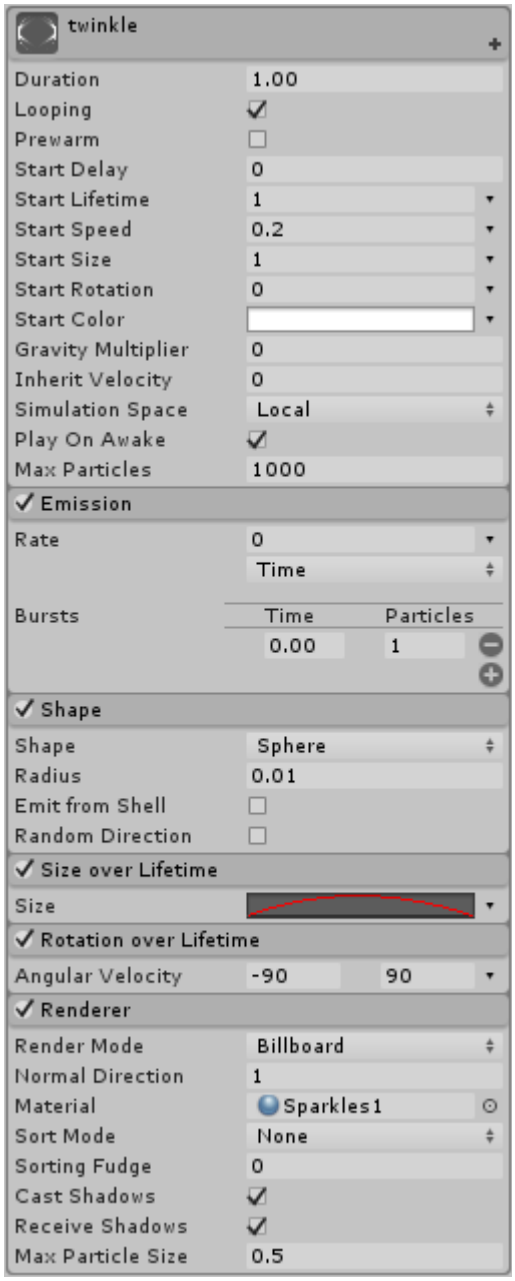
3. Make a new **Particle System**, using the values to the right, and name it twinkle.

- Please keep this particle effect simple! It is designed to be a subtle *accent* that will improve other **Particle System**s soon!

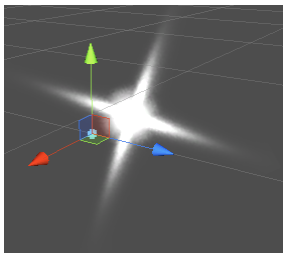
4. **Size over Lifetime** should be enabled, and should use a parabola-like curve (here →).



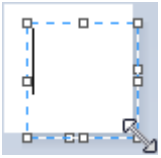
5. **Renderer** should use the **Material**: **Sparkles1** (press ⊙ to find it).



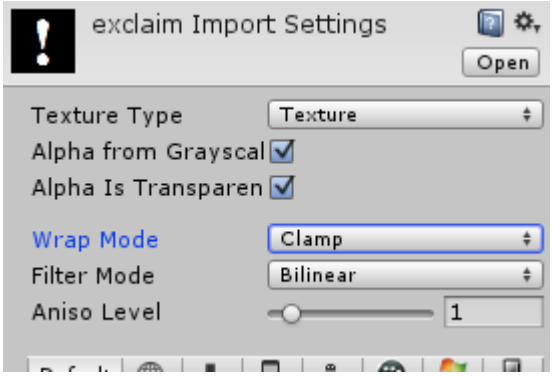
6. **Rotation over Lifetime** should be enabled. Change the **Angular Velocity** value to accepting two values by pressing ▾ and selecting Random Between Two Constants. Use -90 and 90.



7. Particles Using Custom Materials

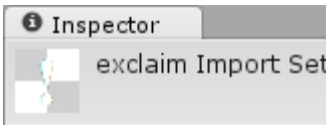


1. Create a new bitmap image using mspaint (explained earlier in this tutorial).
2. Set the image width and height to 64, or another power of two.
3. With the **Text** tool A, click in the image, and expand the text area to fill the height.
4. The sample to the right set text size to 48, font to Georgia, and has a character **!**, in bold.
5. Click outside the text area, or press Home and another tool to finish the **Text** tool.
6. Press **Ctrl + A** to select the entire image, then drag the image with the mouse, or use the arrow keys to center it. Press **Escape** to de-select the image when it is centered.
7. Press **Ctrl + A** again to re-select the entire image (after it was centered). Click on the selection with the **Right-Mouse-Button**, and select Invert color from the menu.
8. Make sure the image is white text (or some white symbol) on a pure black background! Also, make sure the image size is 64 width by 64 height (or another power of 2).



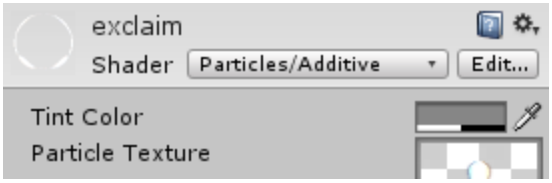
9. **Save** the image and **Import** it into Unity3D.
10. In Unity3D, adjust the image settings in the **Inspector**: Check **Alpha from Grayscale** and **Alpha Is Transparent**. Also, set the **Wrap Mode** to **Clamp**.

- These settings make the image a single stamp that can blend nicely with other particles in a **Particle System**.

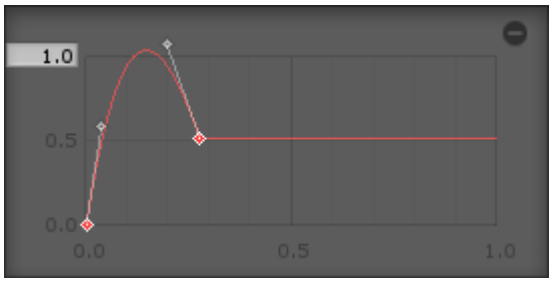


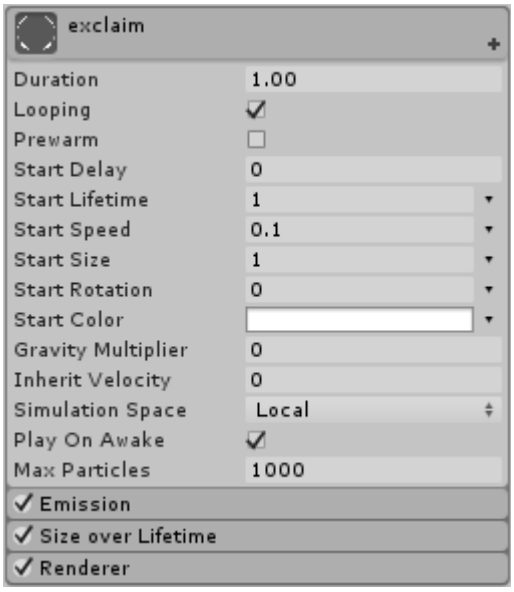
11. Press Apply, and notice that the image icon changed to show black as transparent.
12. Create a new **Material**, and name it exclaim.
13. Set the **Shader** to **Particles** → **Additive**.

- The **Particles/Additive Shader** turns textures into simple maps that can be quickly and efficiently displayed.



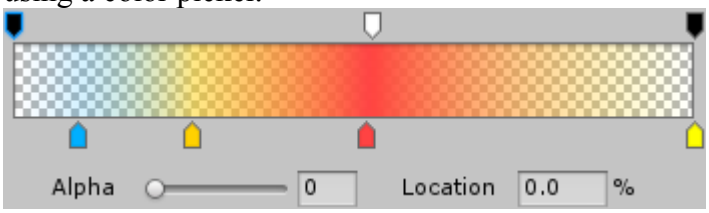
14. Set the **Texture** to exclaim, which was just created and imported.
15. Make a new **Particle System**, using the values to the right, and name it exclaim.
16. *Un-check the* **Shape**.
17. Use the same Emission as twinkle, from the previous section.
18. **Size over Lifetime** should be enabled, and should use a parabola-like curve (here →).
19. **Renderer** should use the **Material**: Exclaim (press ⊙ to find it).



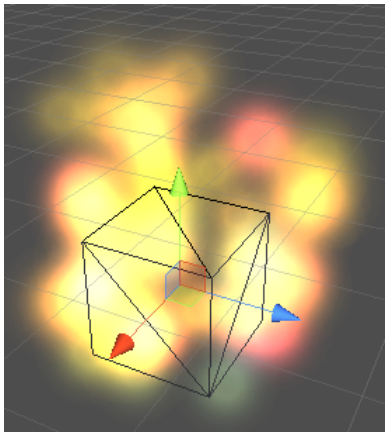


8. Particles With Blending Colors

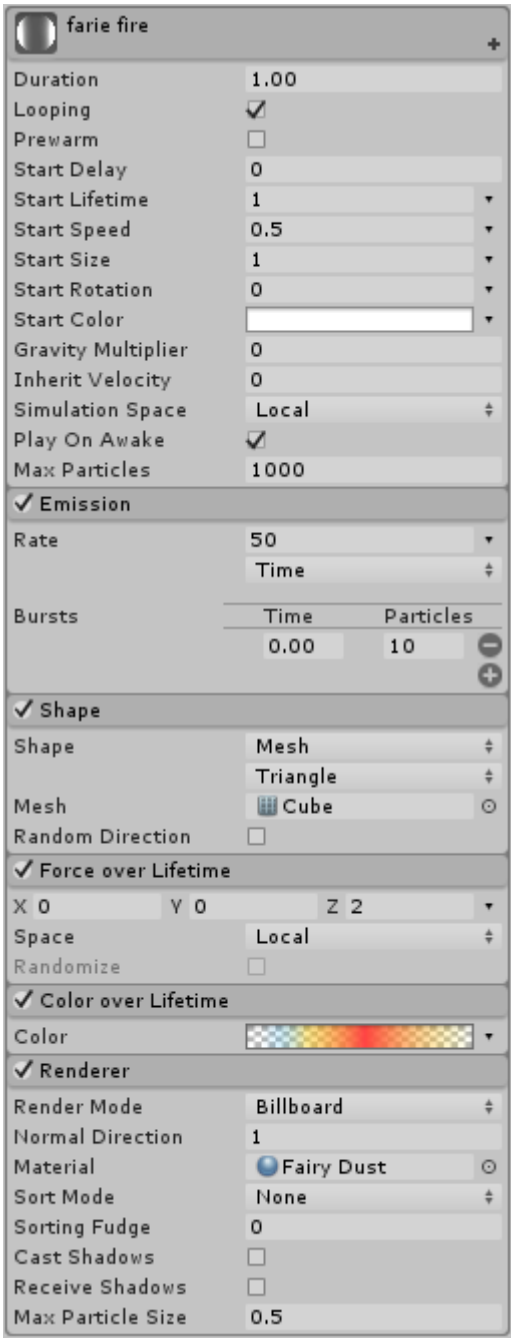
1. Make a new **Particle System**, using the values to the right, and name it "faerie fire".
2. **Color over Lifetime** should be enabled. There should be 4 color *markers* that can be selected, dragged, and modified using a color picker.



3. 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.
4. 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.
5. The color markers can be set to any range that you like.
6. This example uses the **Fairy Dust Material**, but you may want to experiment with other materials to see other effects.

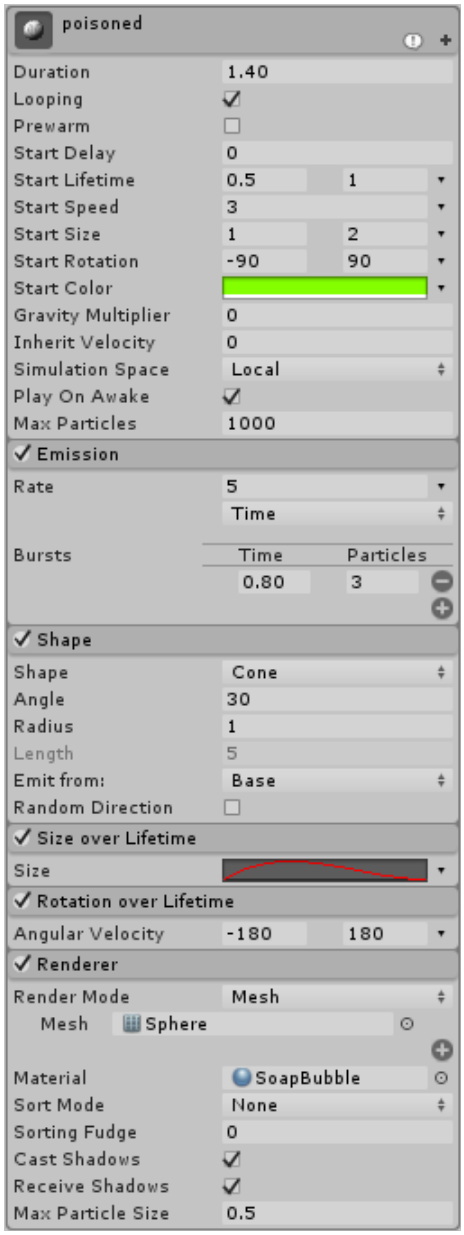


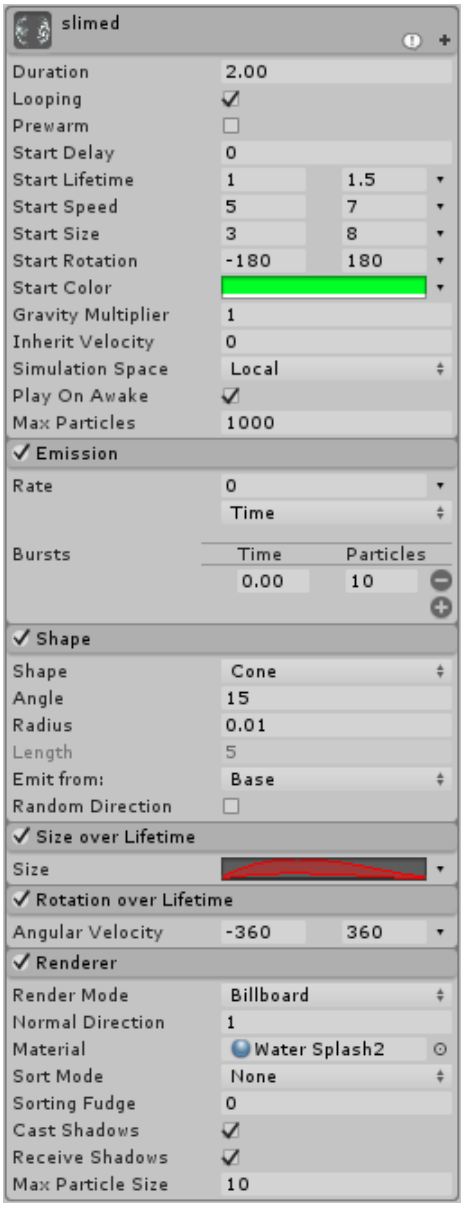
- Note: Be careful with the **Material Editor** that appears below the **Particle Editor** in the **Inspector**! It provides *global access* to the **Material**! That means modifying this material will modify this particle effect, *and* every other object using this material!

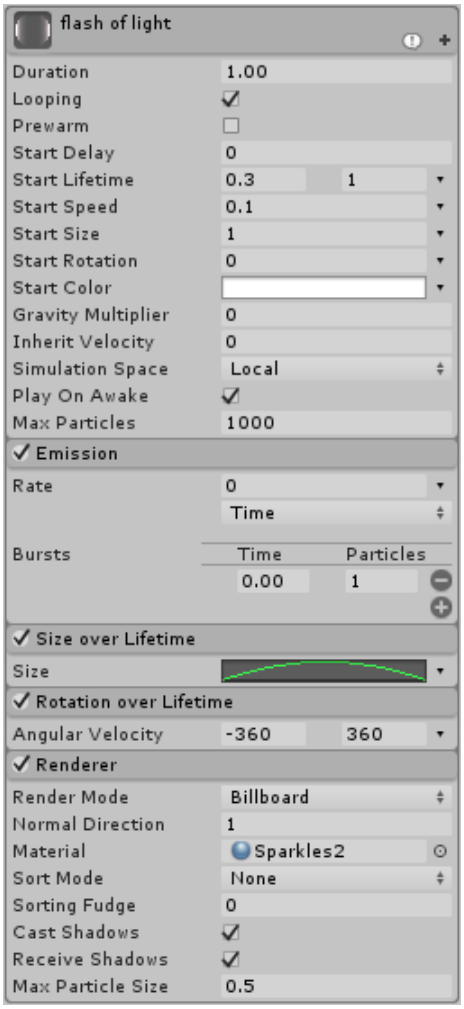


9. More Particle Effects

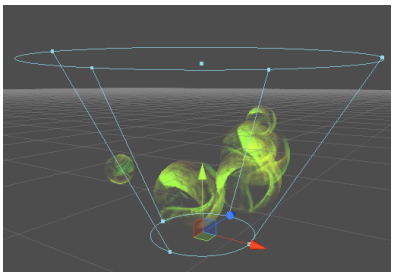
- Try making these **Particle Systems** for practice!



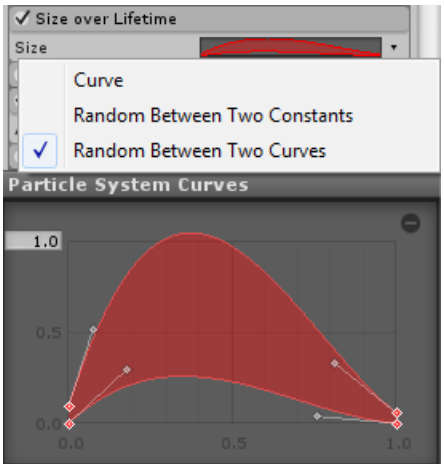


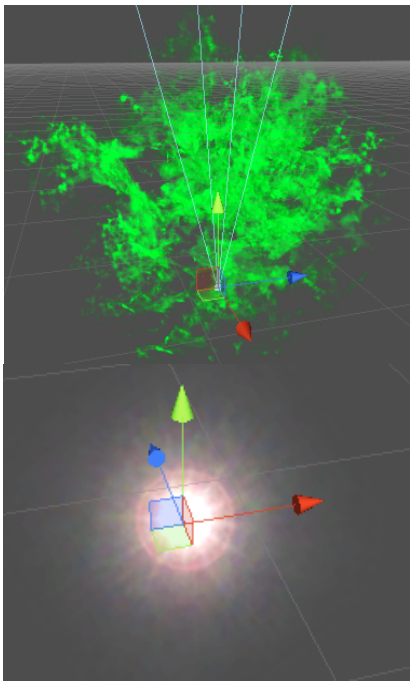


Many of the values that take only 1 number can be adjusted to take more than one number by pressing ▾ and selecting one of the options. This **Particle Effect** makes a lot of use of Random Between Two Constants.



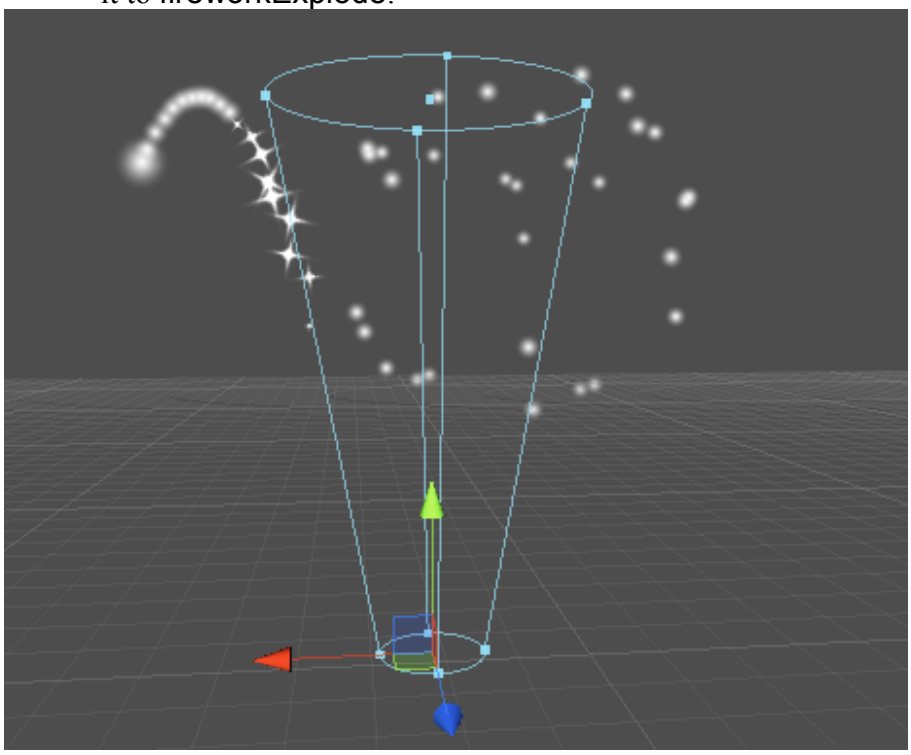
For the slimed particle effect, use 2 curves in **Size over Lifetime**:



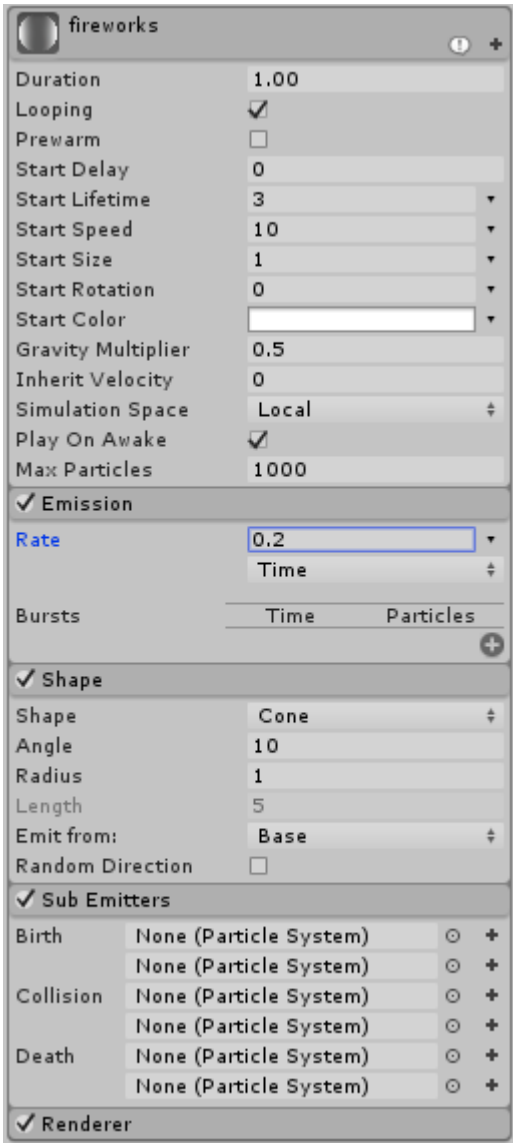


10. Particles With Sub Emitters

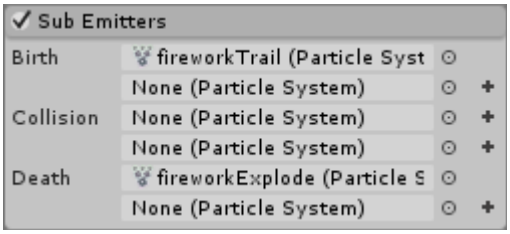
1. Start by creating the simple Particle System to the right →.
2. To the right of **Birth**, in the **Sub Emitters** sub-component, press the + button to add a new birth particle effect.
3. Notice that the fireworks **Particle System** now has a trail for each particle. Also notice in the Hierarchy 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!



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



- The **Material** and **Particle System** are some of the many ways to generate great looking visual effects in Unity3D. Other interesting components you may want to try out include:
  - **3D Text** (at **Hierarchy** → **Create** →)
  - **GUI Text** and **GUI Texture** (at **Hierarchy** → **Create** →)
  - **Trail Renderer** (at **Component** → **Effects** →)