|  |
| --- |
| Continuing Education Program |
| Particle Systems in Unity |

|  |
| --- |
| **Copyright Notice**  Copyright © 2013 DigiPen (USA) Corp. and its owners. All rights reserved.  No parts of this publication may be copied or distributed, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language without the express written permission of DigiPen (USA) Corp., 9931 Willows Road NE, Redmond, WA 98052  **Trademarks**  DigiPen® is a registered trademark of DigiPen (USA) Corp.  All other product names mentioned in this booklet are trademarks or registered trademarks of their respective companies and are hereby acknowledged. |
|  |

1. The Shuriken particle system and you:

Unity comes with an integrated particle system and editor called Shuriken. Although the legacy particle system Unity used to use is still in place, the new Shuriken system is more powerful, but it has the drawback of not being so exposed to scripts.

Before we delve into particle systems we need to distinguish between a Particle System and a Particle Effect.

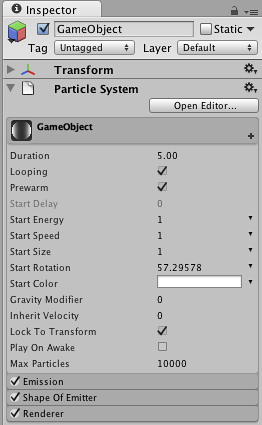
Particle Systems are the building block for a particle effect. A lot of particle effects will use only one particle system, but more complex effects can use more than one particle system. WHEN YOU PLAY AN EFFECT, ALL PARTICLES SYSTEMS IN THAT EFFECT ARE PLAYED.

To create an effect we need several particle systems parented to the same object.

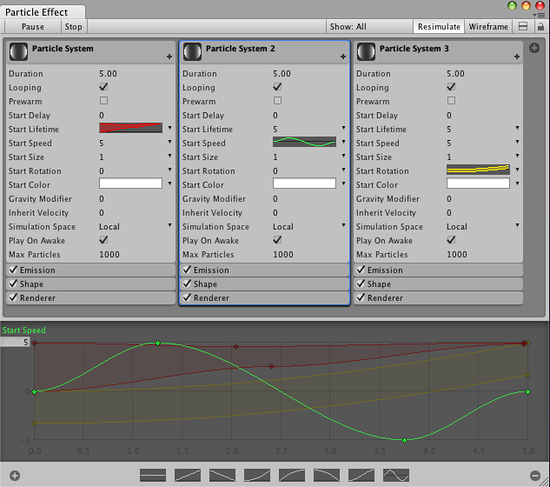
Note: for the purpose of this class, we will continually refer to the Unity project distributed with it (CET\_Particles). When we refer to a specific particle system by name we are talking about that system in the project.

If you look at Particle System 1, you will see it has a child, Particle Child 1. Those two systems create a single effect.

To modify particle systems, we use the particle system editor. To open it, click the “show editor” button on the parent particle system in the inspector.



You will see a window where you can edit each particle system in that effect. You can toggle between “show all” and “show selected” to simulate all systems in the effect or only those ones selected in the scene hierarchy.



Using the editor is the only way to edit whole effects instead of individual systems.

1. Particle systems, modules and values:

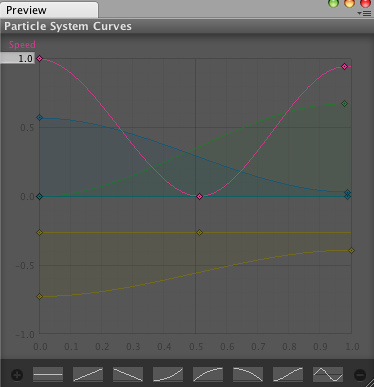
Shuriken particle systems work by applying a set of modules to a particle system. Modules are the little tabs under the particle system in the editor, with the main module being the initial module. Every particle system needs an initial module, a renderer module and an emission module.

Note that when you create a new particle system, it comes with all modules included but disabled. It is good practice to clear all modules and only include those you really need.

Modules allow you to tweak different values of a particle system. Those values can sometimes be switched between constant, random between two constants, curve and random between two curves (which creates a random curve, not parses a random value at each point).

To add a module, click the small + sign in the inspector.

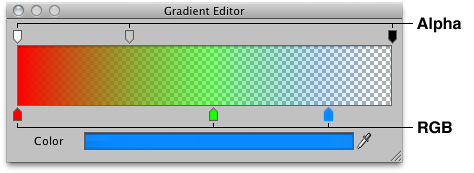
Curves are edited in the curve editor, by adding keys (right mouse button) and moving them around.



Curves can be clicked in the inspector to avoid clutter.

The – button in the curve editor deletes the curve, and the + button parametrizes it (creates a simplified curve with 3 keys, which is faster to process)

Color values are defined as color gradients over lifetime and are edited in its own editor:



There is also to specify a random color gradient between two gradients.

Now we´ll look at the modules one by one and see how each of them works.

1. Initial module:

This module is shared by all particle systems. Cannot be removed.

We will look at the Particle System 1 to explain how this module works. To see the behavior of Particle Child 2 you need to Stop the system and start the simulation again (since it does not loop).

The values this modules allows to tweak are:

Duration: Length of time during which the particle system emits. After this time elapses, it will loop is allowed. ALL CURVES AND GRADIENTS IN THIS MODULE WORK OVER THE DURATION AND NOT THE LIFETIME OF THE PARTICLE.

Looping: Whether the particle system loops (starts emitting from duration equal to 0 after reaching the end of emission).

Prewarm: if the particle system is looping, should it start emitting with no particles present, or as if a full loop had already gone by.

Start Delay: Delay before the particle system starts emitting (and taking into account duration.

Start Lifetime: Life of the particles. How long each is drawn for.

Start Speed: Speed of the particles. How fast they initially move (direction is not relevant here)

Start Size: Size of the particles when emitted.

Start Rotation: Rotation of the particles (billboarded).

Start Color: Starting color.

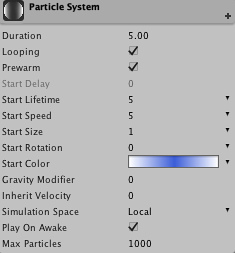
Gravity: If gravity (defined in the physics project prefs or through scripting) affects the particles.

Inherit Velocity: How much velocity should the particles inherit from the parent.

Simulation space: Whether particles are simulated in global or local space (relevant for other modules). If also controls whether moving the system moves the particles in relation to it.

Play On Awake: Should the particle System play when created?

Mac Particles: How many alive particles are permitted to exits at the same time in the system.



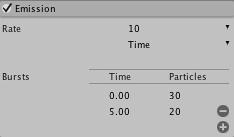
1. Emission Module

This module controls the emission of particles. Without this module, the only way to make a system emit is through scripting.

The values this modules allows to tweak are:

Rate: Emission Rate over time or Distance. If using a curve it uses the DURATION of the system to evaluate the curve.

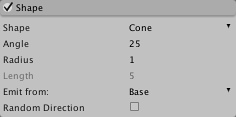
Bursts: Allows to set specific points in the duration of the system to simultaneously emit a set of particles. Good for explosions and other effects.



Look at Particle Effect 2 to see a different effect just by tweaking Particle Effect 1 emission modules.

1. Shape Module

Defines the shape of the emission (and thus the initial direction of particles´ velocities):



Shapes Can be:

-Sphere: Creates particles withitn the volume of a Sphere.

Radius: Radius of the sphere.

Emit From Shell: If checked particles are created only on the surface.

Random direction: If checked, direction of a particle is random. If not, it´s the vector between the center of the sphere and the point of emission.

-HemiSphere: Similar to Shpere but on a hemisphere along the Z axis.

-Cone: Emits on the volume defined by a truncated cone that goes along the Z axis.

Angle: Angle of the cone.

Radius: Radius at the point of emission. A small value creates a non-truncated cone.

Emit from: Base, Base Shell, Volume or Volume Shell.

Length: Length of the volume, if emitting from one.

Random Direction: Otherwise, direction is along the vector defined by the emission position and the vertex of the cone.

-Box: Emits within a box volume.

Box X, Box Y and Box Z: Scale of the Box.

Random direction: Otherwise it will be along the Z axis.

-Mesh: Uses a mesh from emission. Always emits on the surface and never in a volume.

Mesh: The mesh used for emission (it does not assume the mesh assigned to the object is the one to use)

Type: Emit from the vertex, the Triangles or the Edges?

Random Direction: Otherwise the normal at the point of emission.

Look at Particle Effect 3 to see Particle Effect 2 further tweaked to allow for different emission shapes.

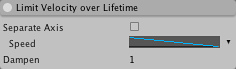
1. Over Lifetime modules:

Now we´ll explore a set of modules that modify particles during EACH PARTICLE LIFETIME. They are very self explanatory, so we´ll explore them together.

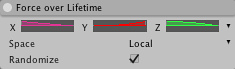
Velocity Over lifetime controls the velocity of the particle along each of the axis of the system OR the global axis.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-3.jpg

Limit velocity over lifetime applies a dampening to the velocity. If has a speed at which the dampening takes place and amount (normalized) of dampening.



Force over Lifetime: Applies a force. Works like velocity except that the force can be randomized each frame (instead of following a random coherent curve).



Color Over Lifetime: Changes the color. Color is MULTIPLIED by whatever initial color the initial module assigned to the particle.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-6.jpg

Size Over Lifetime: Size of the particle. MULTIPLIED by the initial size assigned to the particle by the initial module.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-8.jpg

Rotation over Lifetime: Angular speed of the particle during each lifetime.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-10.jpg

1. By Speed modules

As we can modify particles behavior over their lifetime, we can map those to the current speed of the particle too (in a sense the limit velocity is a mapping of speed on speed).

## All of this modules allow to specify a range at which to clamp the speed for mapping purposes.

These are:

Color by speed: Maps a color to an speed. MULTIPLIED by the initial value.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-7.jpg

Size by Speed: Mapos Size. Multiplied by initial value.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-9.jpg

Rotation to speed: Maps an angular speed to a linear one. OVERWRITES the initial value.

C:\Program Files (x86)\Unity\Editor\Data\Documentation\Documentation\Images\manual\ParticleSystemModules40-11.jpg

Example: Look at Particle Effect 4 for use of by Speed and over lifetime values.

1. External Forces

This module is used only for wind zones. It tells particles to be affected by them.

1. Collision Module

Enables particle collisions.

It allows tweaking of:

Dampen: how to dampen speed after collision.

Bounce: What amount of the particle velocity to set on the normal of the collision.

LifetimeLoss: How much lifetime a particle loses on collision.

Min Kill Speed: at what speed to kill a particle.

Send Messages: Whether to call OnPArticleCollision events to the particleSystem or GameObjects.

Planes/World: Switches between two modes:

Panes:

Planes: transform to use to define a plane. Y axis is used and multiple planes can be defined.

Visualization : How to visualize the planes.

Scale Plane: the scale of the plane.

Particle radius: The assumed radius of the particle to use for collisions.

World: THIS IS EXPENSIVE. USE ONLY FOR LESS THAN1000 PARTICLES IN THE SCENE!!!

Collides With: Collison layers against which to check particles.

Quality: Whther to raycast every particle (high) or just those allotted in the budget (medium) and only every fourth frame (low).

Voxel size: The size of the voxels to catch collisions. Adjust for performance.

1. Sub Emitter.

This module creates a new system PER PARTICLE at different points. Birth, Collision and death.

Useful for stuff like fireworks particles.

1. Renderer Module

This controls the rendering of particcles. Without this module particles will not be rendered.

Parameters:

RenderMode:

Billboard: Always face the camera

Stretched billboard: Streatches the billboard along the camera speed, the particle speed, and on the Y axis.

Horizontal Billboard: Particles align with the XZ axis.

Vertical Billboard: Align with the Y axis and face the camera.

Mesh: particles are rendered on a mesh and not a Quad. A sphjerical mesh is useful for explosions. More expensive.

Normal direction: Whether particles point towards the camera or the center of view. Useful for wide view angle cameras.

Sort Mode: How to sort the particles for render. By distance, youngest first or oldest first.

Sorting fudge: for a particle effect to modify the drawing order of each system.

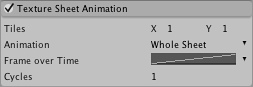
Cast Shadows and receive shadows.

Max Particle Size: Maximum size a particle can have in relation to the camera. Important for performance (see end of the document).

Material: Material used to render the particles. You should always use particle shaders, since they are much faster than regular shaders…

|  |  |
| --- | --- |
|  |  |

1. Texture Sheet Animation.



This is used to animate the particles as if they were sprites. The texture used for the material must be tiled.

Tiles: defines the amount of rows and columns in the texture.

Animation: Whether animate over the whole sheet or just a single row (for random animations).

Whole Sheet:

Frame over lifetime: When to change frame according to the lifetime of the particle.

Single Row:

Random Row: Whether to select a random row for each particle or a specific one (which is memory inefficient, but can be used for simplicity)

Frame Over Time: In the row.

Cycles: Speed of animation, or how many cycles the animation does over the lifetime of the particle.

1. Particle Performance

Particle systems are efficient, but they can carry some costs. The two factors that affect cost are collisions (especially if world collisions are enabled, which should be limited to about 1000 particles on PC) and particle size.

This is because the render cost of a particle (the amount of pixels it renders) is much more important than the simulation cost (which can be negligible). This is more so with transparent particles (which are more costly to render).

A particle system with particles being small on camera does not impact performance, but if that particle is very close to the camera so each particle covers the whole view, it can get very expensive. This is why the max particle size parameter in the renderer is so important.