Particle Emitters

Particle Emitters are a feature in Unreal that create various effects with simulated particles. Throughout the course of my investigation, I discovered what they are in more detail; researched the class components to Particle Emitters; thought of some practical uses; and practiced creating my own effect. My focus is Particle Emitters, but there are plenty of other associated pieces to Particle Effects that can be found on Unreal’s Documentation under the Particle System Reference section [1].

First, I will talk about what Particle Emitters are. Unreal defines them as “a single particle effect that can be held in a ParticleSystem” [2]. This means that when creating a Particle System, the Emitter is what gives the effect to the System that is simulated. Without the Emitter, the System cannot simulate visuals to the create the effect the user is trying to create properly.

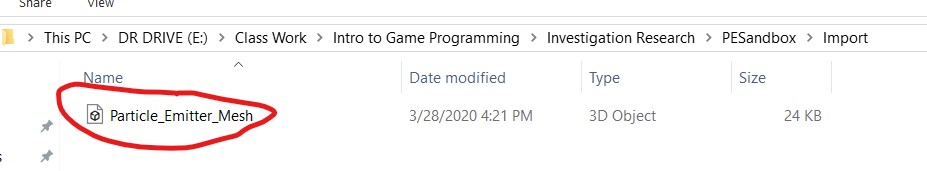
Next, the major class components to a Particle Emitter are as follows: EmitterName, Emitter Render Mode, Emitter Editor Color, InitialAllocationCount, Medium Detail Spawn Rate Scale, and Collapsed [2]. EmitterName is the name the user gives to the emitter. Emitter Render Mode contains several settings: ERM\_Normal, ERM\_Point, ERM\_Cross, and ERM\_None. The Normal setting renders the particles as they were intended. Point renders each particle as a 2x2 pixel block without scaling and uses the emitter’s editor color [2]. Cross renders the particles as a cross of lines scaled by any size modules and uses the emitter editor color. Finally, None simply does not render any particles [2].

Emitter Editor Color affects the color of the particle emitter block when it is collapsed, and also affects this in the Curve Editor and debug rendering nodes as well. InitialAllocationCount allows the user to declare the initial number of particles allocated at emitter initialization. If this is set to 0, the calculated peak amount is used instead [2]. The Medium Detail Spawn Rate Scale is used to scale down the spawn rate of the particle emitter while Unreal is in medium or low detail mode, and can be used to optimize the draw cost in splitscreen mode. If this is set to 0, the particle emitter is not active when Unreal is not in high detail mode [2]. Finally, Collapsed is a boolean that determines whether the Particle Emitter will appear collapsed in the Emitter List in cascade. To toggle this property, double clicking on the emitter will work [2].

Next, I will cover some ideas I thought of that are practical for a particle emitter. One example I thought of is using the explosion, fire, and smoke particles Unreal has pre-created to make an exploding item. Another example is putting flames on a pyre. In the tutorial I watched, the lecturer created a trailing effect for a spaceship [3]. I was unable to create this exactly because I did not have access to the files he used to create this effect, but I did create my own effect similar to the lecturer’s with what I had access to. There are many other examples of uses for particle emitters, but these are a few that I thought of or have seen in my research.

Finally, I will go over the work I did along with the tutorial. Because I did not have access to the files the lecturer used, I created my own like the project he worked on. I figured out several things at this point: I discovered how to export files from the animation software Maya into Unreal, I learned how to implement Particle Emitters in a very basic way, and I figured out how to add them to a Blueprint and integrate changes to the emitters in the level editor.

The first thing I did was go into Maya to create a cone shape to use. To do this, it was a simple task of bringing a cone and modifying it to generally match the cone the lecturer had in his project. Next, I exported it from Maya into Unreal. To get this to work, I had to also go into Unreal and import the file I exported. This was trivial from both Maya and Unreal. In Maya, it goes as follows: File > Send to Unreal and then choose the project to Export the file to. In Unreal, it was as follows: File > Import and then select the file to import.



Next, I went into Unreal and went along with the tutorial as best as I was able. I had to do quite a bit to make the particle effects work in unison with the cone but was successful after some experimentation to produce something effective. I will talk about the material I made for the cone, then the blueprint, then the particle system and its materials. For each explanation, I will be explaining the process from left to right in the Blueprint to make the translation easier.

First, I will discuss the material I made for the cone. Starting with the effect speed, I used the values the lecturer used for constant speed values and the default MainSpeed [3]. A screenshot of a computer

Description automatically generated

I then attached a panner to attach the speed to the UV Map for the texture. Next, I attached the Texture Sample to a Mask, taking the R and G values of the UV Map. I then multiplied these by a constant value of .05 and added them to the Texture Coordinates. This value was then sent into a panner, along with another speed constant. I then took two Texture Samples of the Fire Texture, one of which had a speed constant multiplied by -0.15 (to simulate the speed differently) and sent them into a multiply node.

A picture containing black, parking, sign, standing

Description automatically generated

This node took the R value of each (which made the texture show up better than with the R and G values that the lecturer used)[3]. From this point, I multiplied the two values by a constant 4, used a Saturate node (which is similar to a clamp node between 0 and 1 for particle effects specifically) and used a Power node with an exponent of 4 [3]. I then gathered the Texture Coordinates, attached them to a Mask node for the R and G values, and multiplied them together and sent them in for the Opacity. Finally, I added a Color node, multiplied it by 12, and sent this in for the Emissive Color.

A close up of a computer

Description automatically generated

Next, I created an Actor Blueprint to further follow along with the tutorial and experiment on my own. The lecturer did several things here in the viewport: he added the cone mesh, added a particle system, and added a point light [3]. Then, in the construction script, he added several variables to be manipulated in the level editor. He also made use of several functions to be able to manipulate the variables of the particle emitter and material. This is the point at which I was able to follow along best because it was a trivial matter of editing a Blueprint.

A close up of a computer

Description automatically generatedA close up of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

To begin, I made the MyMesh component (the cone I added to the blueprint) a Dynamic Material Instance based on the material I created previously. Next, I set the Exhaust Material variable to this and use the Set Vector Parameter Value node to be able to change the Color of the material. After this, I used a Set Color Parameter node to be able to do the same thing for the Particle Effect I called My Exhaust. I then used a Set Light Color node to change the color of the variable I called My Light. Then, I used the node Set Scalar Parameter Value to adjust the MainSpeed of the Particle Emitter. Finally, I used the node Set Float Parameter to adjust the SmokeEmit parameter of the Particle Emitter. I reiterate that this is all the process that the lecturer used in the tutorial as well [3].

Finally, I will discuss the actual Particle Emitter system I created based on the tutorial. This was also relatively close to what the lecturer did as well because I had access to the resources he used because they are starter content in Unreal [3]. I did however adjust some values in my project file to better showcase the work I did and make it easier to see. There are two effect components to the Particle System I made: Sparks and Smoke. Each are named exactly for the effect they simulate. In the Sparks, the components I adjusted were the Initial Size, the Initial Velocity, the Color Over Life, and the Scale Color/ Life. This order is also pictured below to show the settings that they had at the end.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

I also had a sparks material created (following with the tutorial) to adjust the particle effect slightly [3].

A picture containing indoor, black, sitting, small

Description automatically generated

Next, for the Smoke portion of the Particle Effect, I added a Subimage Index, a Size by Life, and a Drag. Because the Smoke effect uses a SubUV to produce the simulated particles, it requires the Subimage index settings shown below. I also adjusted the settings of the Size By Life piece and the Drag.

A screenshot of a cell phone

Description automatically generated

A white sign with black text

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Finally, I also created the material for the Smoke effect.

A screenshot of a computer

Description automatically generated

Again, this was all following along with the tutorial [3].

Particle Effects are interesting to create and manipulate. They have a lot of working parts to work with in order to create some amazing effects. They require a lot more than I initially believed they would, but the reward is well worth the effort.

# Bibliography

|  |  |
| --- | --- |
| [1] | Epic Games Inc., "Particle System User Guide," [Online]. Available: https://docs.unrealengine.com/en-US/Engine/Rendering/ParticleSystems/UserGuide/index.html. [Accessed 27 January 2020]. |
| [2] | Epic Games Inc., "Particle Emitter Class," [Online]. Available: https://docs.unrealengine.com/en-US/Engine/Rendering/ParticleSystems/Reference/BaseClasses/ParticleEmitterClass/index.html. [Accessed 27 January 2020]. |
| [3] | Epic Games Inc., "VFX and Particle Systems with Unreal Engine Tutorial," [Online]. Available: https://learn.unrealengine.com/course/2436530?r=False&ts=637157479411418523. [Accessed 27 January 2020]. |