PARTICLE SYSTEM MODULES – FAQ
KARL JONES, APRIL 20, 2016
Starting with Unity 5.3, you have full scripting access to Particle System modules. When you use a Particle System module, it is important to understand how the curves and constants are handled. Here, we’ll take a look at how you can access these properties and copy them between systems.

### Emission Module

In an emission module, you can access the `rate` property, which is a float that determines how many new particles are generated per second. If you want to get the emission module, you can use the following code:

```csharp
ParticleSystem.EmissionModule emission = myParticleSystem.emission;
```

This will return an instance of the `ParticleSystem.EmissionModule` class, allowing you to access properties such as `rate` and `constantMin`.

#### Example

Let’s walk through an example where we create a simple linear curve and then apply it to the emission module:

```csharp
// Get the emission module.
ParticleSystem.EmissionModule emissionModule = myParticleSystem.emission;

// A simple linear curve.
AnimationCurve linearCurve = new AnimationCurve();
linearCurve.AddKey(0.0f, 0.0f);
linearCurve.AddKey(1.0f, 1.0f);

// Assign the curve to the emission module
emissionModule.rate = linearCurve;
```

### ParticleModuleExample

`ParticleModuleExamples` is a script that demonstrates how to access and modify curves. It contains an example where we create a curve and apply it to different parts of the particle system. This script also shows how to copy properties from one system to another, making your workflow more efficient.

#### Code Snippet

```csharp
// Get the emission module.
ParticleSystem.EmissionModule emissionModule = myParticleSystem.emission;

// Get the system and the emission module.
ParticleSystem particleSystem = GetComponent<ParticleSystem>();
ParticleSystem.EmissionModule emissionModule = particleSystem.emission;

// Set the rate of the emission module.
emissionModule.rate = linearCurve;
```

### MinMaxCurve

In Unity, you can use the `MinMaxCurve` class to create a curve that has a minimum and maximum value. This can be useful when you want to limit the range of the curve to a specific interval.

#### Example

```csharp
// Create a MinMaxCurve.
MinMaxCurve ourCurve = new MinMaxCurve();
ourCurve.AddKey(0.0f, 0.0f);
ourCurve.AddKey(1.0f, 1.0f);
```

### Curve Access

Curves are optimized to improve performance, and they are easily accessible through the `minMaxCurve` property of the `ParticleSystem` class. To access the `minMaxCurve` property, you can use the following code:

```csharp
ParticleModuleExamples myParticleSystem = GetComponent<ParticleSystem>();
MinMaxCurve ourCurveMin = myParticleSystem.minMaxCurve;
```

### Conclusion

In conclusion, Unity’s Particle System modules provide a rich set of tools for creating dynamic particle effects. With full scripting access, you can customize the behavior of your particles to create engaging and immersive experiences.

---

This tutorial is part of our ongoing series exploring Unity’s Particle System capabilities. Stay tuned for more insights and examples to help you master this powerful feature.