Creating Particle Effects

Add particle effects to your app by creating repeatable particles in Xcode's editor, or in code.

Overview

Add special effects to your app, such as glittering or realistic fire with smoke, using particle effects. You create and configure particle effects using Xcode's SpriteKit Particle Editor, or in code. A benefit of using the editor is that you can repeat the particles in different places across your app without duplicating the code.

Create Emitters with the Particle Editor

In most cases, you never need to configure an emitter node directly in your game. Instead, you use Xcode to configure an emitter node's properties. As you change the behavior of the emitter node, Xcode immediately provides you an updated visual effect. When complete, Xcode archives the configured emitter. Then, at runtime, your game uses this archive to instantiate a new emitter node.

Using Xcode to create emitter nodes has a few important advantages:

- >> It is the best way to learn the capabilities of the emitter class.
- >> You can quickly experiment with new particle effects and see the results immediately.
- >> You separate the task of designing a particle effect from the programming task of using it. Your artists can work on new particle effects independent of your game code.

The following code shows how to load a particle effect that was created by Xcode. All particle effects are saved using Cocoa's standard archiving mechanisms, so the code first creates a path to the smoke effect, and then loads the archive.

```
func newSmokeEmitter() -> SKEmitterNode? {
    return SKEmitterNode(fileNamed: "smoke.sks")
}
```

Configure Particles in Code

The SKEmitterNode class provides many properties for configuring an emitter node's behavior. The Xcode emitter editor sets the same property values. You can also create and configure your own emitter, or you can take an emitter node created in the Particle Emitter Editor, load it, and change its property values.

When the emitter node is in a scene, it emits new particles. You use the following properties to define how many particles it creates:

>> The particleBirthRate property specifies the number of particles that the emitter creates every second. For example, consider using the smoke effect in the previous code to show damage to a rocket ship. As the ship takes more damage, you can increase the birth rate of the emitter to add more smoke.

>> The numParticlesToEmit property specifies how many particles are created before the emitter turns itself off. You can also configure the node to emit an unlimited number of particles.

When a particle is created, its initial property values are determined by the properties of the emitter. For each of the particle's properties, the emitter class declares up to four properties:

>> The average starting value for the property.

>> A random range for values of the property. Each time a new particle is emitted, a new random value is calculated within that range.

>> The rate at which the value changes over time, also known as the property's speed. Not all properties have a speed property.

>> An optional keyframe sequence.

The complete list of properties used to configure an emitter node is given in SKEmitterNode.

The following code shows how you might configure an emitter's scale property. This is a simplified version of a node's xScale and yScale properties, and determines how large the particle is.

```
myEmitter.particleScale = 0.3;
myEmitter.particleScaleRange = 0.2;
myEmitter.particleScaleSpeed = -0.1;
```

When a new particle is created, its scale value is a random number from 0.2 to 0.4. The scale value then decreases at a rate of 0.1 per second. So, for example, if a particular particle started at the average value, 0.3, it would decrease from 0.3 to 0 over a period of 3 seconds.