Getting Started with Sprite Nodes

Learn the basics about using images, also known as sprites, with SpriteKit.

Overview

The simplest way to create a textured sprite node is to have SpriteKit create both the texture and the sprite for you. You store the artwork in the app bundle (ideally in an asset catalog), and then load it at runtime, as shown in this code:

```
let spaceship = SKSpriteNode(imageNamed: "rocket.png")
spaceship.position = CGPoint(x: 100, y: 100)
self.addChild(spaceship)
```

Default Behavior

When you create a sprite as shown in the code above, you get a lot of default behavior for free:

>> The sprite node is created with a frame that matches the texture's size.

>> The node is rendered so that it is centered on its position. The node's frame property holds the rectangle that defines the area it covers.

>> The texture is alpha blended into the frame buffer.

>> An SKTexture object is created and attached to the node. This texture object automatically loads the texture data whenever the sprite node is in the scene, is visible, and is necessary for rendering the scene. Later, if the sprite is removed from the scene or is no longer visible, SpriteKit can delete the texture data if it needs that memory for other purposes. This automatic memory management simplifies but does not eliminate the work you need to do to manage art assets in your game.

Textures

Although SpriteKit can create textures for you automatically when a sprite is created, in more complex apps you need finer control over textures. For example, you might want to do any of the following:

>> Share a texture between multiple sprites.

>> Change a sprite's texture after it is created.

>> Animate a sprite through a series of textures.

>> Create textures from data that is not directly stored in the app bundle.

>> Render a node tree into a texture. For example, you might want to take a screenshot of your gameplay to show to the player after he or she completes the level.

>> Preload textures into memory before presenting a scene.

You do all of these things by working directly with texture objects. For more information, see the SKTexture class reference.

Customized Rendering Stages

You can use each sprite node’s properties to independently configure four distinct rendering stages:

>> Move a sprite node’s frame so that a different point in the texture is placed at its position. See Using the Anchor Point to Move the Sprite Node’s Frame.

>> Resize a sprite node. You control how the texture is applied to the sprite when the size of the sprite does not match the size of the texture. See Resizing a Sprite in 9 Parts.

>> Colorize a sprite node’s texture when it is applied to the sprite. See Colorizing a Sprite Node.

>> Use other blend modes in a sprite node to combine its contents with that of the framebuffer. Custom blend modes are useful for lighting and other special effects. See Blending the Sprite into the Frame Buffer.

Art and Property Configuration Matching

Often, configuring a sprite node to perform these four steps—positioning, sizing, colorizing, and blending—is based on the artwork used to create its texture. This means that you rarely set property values in isolation from the artwork. You work with your artist to ensure that your game is configuring the sprites to match the artwork.

Here are some of the possible strategies you can follow:

>> Create the sprite nodes with hardcoded values in your project. This is the fastest approach, but the least desirable in the long term, because it means that the code must be changed whenever the art assets change.

>> Create your own tools using SpriteKit to fine tune the sprite's property values. When you have a sprite node configured the way you want it, save the sprite to an archive. Your game uses the archive to create sprites at runtime.

>> Store the configuration data in a property list that is stored in your app bundle. When the sprite node is loaded, load the property list and use its values to configure the node. Your artist can then provide the correct values and change them without requiring changes to your code.