About Texture Atlases

Learn about the benefits of having a texture atlas, and the process for using it.

Overview

In most cases, a SpriteKit texture atlas is created automatically by the Xcode build system after you put multiple images in an .atlas file in the app bundle. Then, you access those images at runtime using `textureNamed(_:)`.

The benefits are:

1. Minimized bundle size.
2. Maximized runtime performance when rendering more than one of the atlas's textures at the same time, because SpriteKit is able to effectively render them at the same time.

Creating a Texture Atlas

You create texture atlases using Xcode. Add a folder with the .atlas extension to your project and place the artwork to be included in the atlas inside this folder. Xcode builds a texture atlas from the artwork and adds it to the app bundle. The format for a compiled texture atlas is private and subject to change.

Loading Textures from a Texture Atlas

Creating a texture using `init(imageNamed:)` will create a new `SKTexture` object from an image file in a texture atlas if one is available.

SpriteKit searches first for an image file with the specified filename. If it doesn't find one, it searches inside any texture atlases built into the app bundle. This means that you don't have to make any coding changes to support this in your game. This design also offers your artists the ability to experiment with new textures without requiring that your game be rebuilt. The artists drop the textures into the app bundle. When the app is relaunched, SpriteKit automatically discovers the textures (overriding any previous versions built into the texture atlases). When the artists are satisfied with the textures, you then add those textures to the project and bake them into your texture atlases.

If you want to explicitly work with texture atlas objects, use the SKTextureAtlas class. First, create a texture atlas object using the name of the atlas. Next, use the names of the image files stored in the atlas to look up the individual textures. The following code shows an example. It uses a texture atlas that holds multiple frames of animation for a monster, and shows how to create textures from those frames and store them in an array. In the actual project, you would add a monster.atlas folder with the four image files.

```
let atlas = SKTextureAtlas(named: "monster")
let f1 = atlas.textureNamed("monster-walk1.png")
let f2 = atlas.textureNamed("monster-walk2.png")
let f3 = atlas.textureNamed("monster-walk3.png")
let f4 = atlas.textureNamed("monster-walk4.png")
let monsterWalkTextures = [f1, f2, f3, f4]
```

Whether your project uses a texture atlas or a sprite atlas, the code used to load assets is identical.

Creating a Texture Atlas at Runtime

Provide a dictionary that points to the various image files that should be part of the atlas. The following code shows how you can create an SKTextureAtlas object using the init(dictionary:) from three image files. The initializer doesn't allow for optional values in the dictionary, so a guard statement is used to ensure that all of the images exist.

```
guard let pandaImage = UIImage(named: "panda.png"),
    let monkeyImage = UIImage(named: "monkey.png"),
    let rabbitImage = UIImage(named: "rabbit.png") else {
        print("unable to resolve all images")
        return
}
let textureAtlas = SKTextureAtlas(dictionary: ["panda": pandaImage,
                                               "monkey": monkeyImage,
                                               "rabbit": rabbitImage])
```

Creating a Sprite Atlas

A sprite atlas offers the advantages of a texture atlas with the management functionality of an asset catalog. For example, if you supply assets with different resolutions for different devices, sprite atlases support app thinning, and your users will only download the required images.

You create a sprite atlas using Xcode. First add a new asset catalog to your project. Then, inside the new asset catalog, add a new sprite atlas.

In this document, the term texture atlas can refer to either a sprite atlas or a texture atlas.

Deciding on Texture Count and Size

To get the most out of a texture atlas, pack it full of textures you'll be using simultaneously. But be aware that you need a balance between too many textures and too few.

> >> If you include too many images, large amounts of texture data may need to be loaded into memory simultaneously.
>
> >> With a common maximum texture size of 4,096 x 4,096 pixels, the build system will ideally fit all of your source textures into one destination texture. The build system may need to disperse your source textures across multiple destinations textures depending on the count and size of your source textures.
>
> >> If you use too few images, you won't be getting the most out of the texture atlas's batch rendering ability.

Experiment with different configurations of your texture atlases and choose the combination that gives you the best performance. Xcode builds the atlases for you, so you can switch between different atlas configurations with relative ease.