Preloading Textures into Memory

Decompress images ahead of time to avoid performance issues during gameplay.

Overview

A major advantage to SpriteKit is that it performs a lot of memory management for you automatically. When rendering a new frame of animation, SpriteKit determines whether a texture is needed to render the current frame. If a texture is needed but is not prepared for rendering, SpriteKit loads the texture data from the file, transforms the data into a format that the graphics hardware can use, and uploads it to the graphics hardware.

This process happens automatically in the background, but it isn't free. If too many unloaded textures are needed at once, it may be impossible to load all the textures in a single frame of animation, causing the frame rate to stutter. To avoid this problem, you need to preload textures into memory, particularly in larger or complex games.

Preload Texture Objects

This code shows how to preload an array of SKTexture objects. The preload(_:withCompletionHandler:)method calls the completion handler after all of the textures are loaded into memory. In this example, all of the textures for a particular level of the game are preloaded in a single operation. When the textures are all in memory, the completion handler is called. It creates the scene and presents it. (You need to add code to provide these texture objects to the scene; that code isn't shown here).

```
SKTexture.preload(textureArrayForLevel1) {
    // The textures are loaded into memory. Start the level.
    let gameScene = GamePlayScene(size: CGSize(width: 768, height: 1024))

    if let spriteView = view as? SKView {
        spriteView.presentScene(gameScene)
    }
}
```

Choose a Time to Preload

Because you are intimately familiar with the design of your game or app, you are the best person to know when new textures are needed. The exact design of your preloading code is going to depend on your game engine. Here are a few possible designs to consider:

>> For a small game or app, you may be able to preload all of its textures when the app is launched, and then keep them in memory forever.

>> For a larger game or app, you may need to split the textures into levels or themes. Each level or theme's textures are designed to fit in a specific amount of memory. When the player starts a new level, you preload all of that level's texture objects. When the player finishes playing the level, the textures not needed for the next level are discarded. When you preload levels, the load time is all up front, before gameplay starts.

>> If a game needs more textures than can fit into memory, you need to preload textures dynamically as the game is being played. Typically, you preload some textures at the start of a level, and then load other textures when you think they will be needed soon. For example, in a racing game, the player is always moving in the same direction, so for each frame you might fetch a new texture for content the player is about to see. The textures are loaded in the background, displacing the oldest textures for the track. In an adventure game that allows for player-controlled movement, you might have to provisionally load textures when a player is moving in a particular direction.