Searching the Node Tree

Access nodes by name to avoid needing an instance variable.

Overview

The nodes in the scene tree are often organized to determine the precise rendering order for the scene, not the role these nodes play in your scene. You may also be loading nodes from an archive file rather than instantiating them directly at runtime, so you may need to be able to find specific nodes within a node tree. To do this, you provide names to nodes and then search for those names. The node name usually serves two purposes in your app:

>> You can write your own code that implements game logic based on the node's name. For example, when two physics objects collide, you might use the node names to determine how the collision affects gameplay.

>> You can search for nodes that have a particular name. Typically, this is done once, when a scene is first loaded.

Name Your Node

A node's `name` property should be an alphanumeric string without any punctuation. The following code shows how you might name three different nodes to distinguish them from each other.

```
playerNode.name = "player"
monsterNode1.name = "goblin"
monsterNode2.name = "ogre"
```

When you name nodes in the tree, decide whether those names will be unique. If a node's name is unique, you should never include more than one node with that name in the scene tree. On the other hand, if a node name is not unique within your app, it might represent a collection of related nodes. For example, in the code above, there are probably multiple goblins within the game, and you might want to identify them all with the same name. But the player might be a unique node within the game.

Use a Simple Search

The SKNode class implements the following methods for searching the node tree:

- The `childNode(withName:)` method searches a node's children until it finds a matching node, then it stops and returns this node. This method is usually used to search for nodes with unique names.

- The `enumerateChildNodes(withName:using:)` method searches a node's children and calls your block once for each matching node it finds. You use this method when you want to find all nodes that share the same name.

- The `subscript(_:)` method returns an array of nodes that match a particular name.

The following code shows how you might create a method on your scene class to find the player node. You might use a method like this inside your code to load and prepare a scene.

```
var playerNode: SKNode? {
    return childNode(withName: "player")
}
```

When this method is called on the scene, the scene searches its children (and only its children) for a node whose name property matches the search string, then returns the node. When specifying a search string, you can either specify the name of the node or a class name. For example, if you create your own subclass for the player node and name it PlayerSprite, then you could specify PlayerSprite as the search string instead of player; the same node would be returned.

Use an Advanced Search

The default search only searches a node's children and must exactly match either the node's name or its class. However, SpriteKit provides an expressive search syntax so that you can perform more advanced searches. For example, you could do the same search as before but search the entire scene tree. Or you could search the node's children, but match a pattern rather than requiring an exact match.

Table 1 describes the different syntax options. The search uses common regular expression semantics.

TABLE 1 SEARCH SYNTAX OPTIONS

Syntax	Description
/	When placed at the start of the search string, this indicates that the search should be performed on the tree's root node. When placed anywhere but the start of the search string, this indicates that the search should move to the node's children.
//	When placed at the start of the search string, this specifies that the search should begin at the root node and be performed recursively across the entire node tree. Otherwise, it performs a recursive search from its current position.
.	Refers to the current node.
..	The search should move up to the node's parent.
*	The search matches zero or more characters.
[characters delimited by commas or dashes]	The search matches any of the characters contained inside the brackets.
alphanumeric characters	The search matches only the specified characters.

Table 2 shows some useful search strings to help get you started.

TABLE 2 EXAMPLE SEARCHES

Search string	Description
MyName	Matches any of the current node's children that are named MyName.
/MyName	Matches any of the root node's children that are named MyName.
//MyName	Matches any nodes in the tree that are named MyName.
.//MyName	Matches any descendants of the current node that are named MyName.
//MyName/..	Finds the parent nodes of any nodes in the tree that are named MyName.
//*	Matches every node in the tree.
A[0-9]	Matches any of the current node's children that are named A0, A1, …, A9.
Abby/Normal	Matches any grandchild of the current node when the grandchild is named Normal and the grandchild's parent is named Abby.
//Abby/Normal	Matches any nodes in the tree that are named Normal and whose parents are named Abby.