About Node Drawing Order

Understand how SpriteKit layers your scene's nodes from top to bottom.

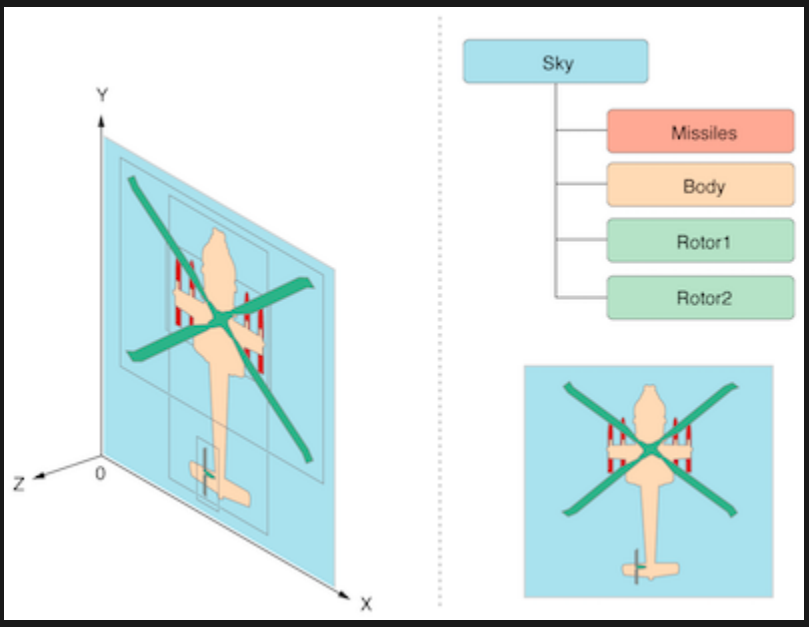
Overview

The standard behavior for scene rendering follows a simple pair of rules (as shown in Figure 1):

>> A parent draws its content before rendering its children.

>> Children are rendered in the order in which they appear in the child array.

FIGURE 1 PARENTS DRAWN BEFORE CHILDREN



In the figure above, the helicopter body and its components are all children of the sky node. So the scene renders its content as follows:

1. The scene renders itself, clearing its contents to its background color.
2. The scene renders the sky node.
3. The sky node renders its children—the helicopter body and its components—in the order they were added as children.

Draw Order Relative to the Parent Node

Maintaining the order of a node's children can be a lot of work. Instead, you can give each node an explicit height in the scene. You do this by setting a node's `zPosition` property. The zPosition is the node's height relative to its parent node, much as a node's `position` property represents its x and y position relative to its parent's position. So you use zPosition to place a node above or below the parent's position.

When you take zPosition into account, here is how the node tree is rendered:

1. Each node's global zPosition is calculated by recursively adding its zPosition to its parent's zPosition.
2. Nodes are drawn in order from smallest zPosition to largest zPosition.
3. If two nodes share the same zPosition, ancestors are rendered first, and siblings are rendered in child order.

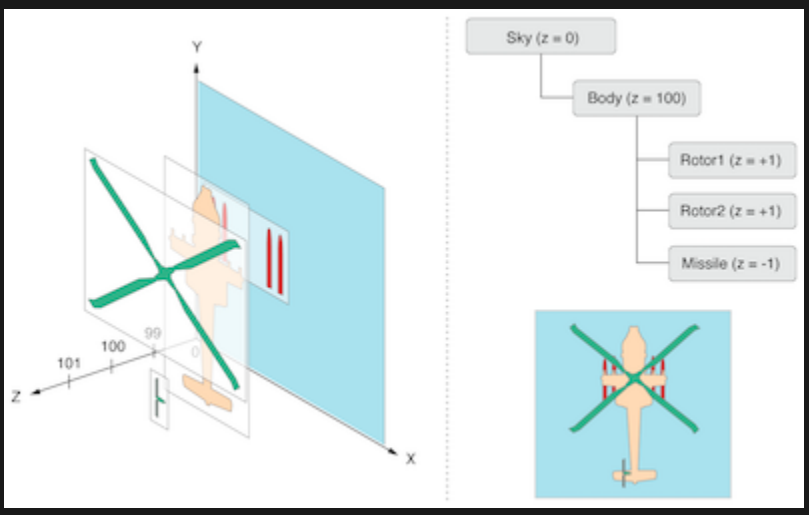
Sibling Order Performance

SpriteKit uses a deterministic rendering order based on the height nodes and their positions in the node tree. But, because the rendering order is so deterministic, SpriteKit may be unable to apply some rendering optimizations that it might otherwise apply. For example, it might be better if SpriteKit could gather all of the nodes that share the same texture and drawing mode and draw them with a single drawing pass. To enable these sorts of optimizations, you set the view's `ignoresSiblingOrder` property to true.

When you ignore sibling order, SpriteKit uses the graphics hardware to render the nodes so that they appear sorted by zPosition. It sorts nodes into a drawing order that reduces the number of draw calls needed to render the scene. But with this optimized drawing order, you cannot predict the rendering order for nodes that share the same height. The rendering order may change each time a new frame is rendered. In many cases, the drawing order of these nodes is not important. For example, if the nodes are at the same height but do not overlap on screen, they can be drawn in any order.

Figure 2 shows an example of a tree that uses zPosition to determine the rendering order. In this example, the body of the helicopter is at a height of 100, and its children are rendered relative to its height. The two rotor nodes share the same height but do not overlap.

FIGURE 2 DEPTH-ONLY RENDERING CAN IMPROVE PERFORMANCE



Interleaved Child Nodes from Different Parents

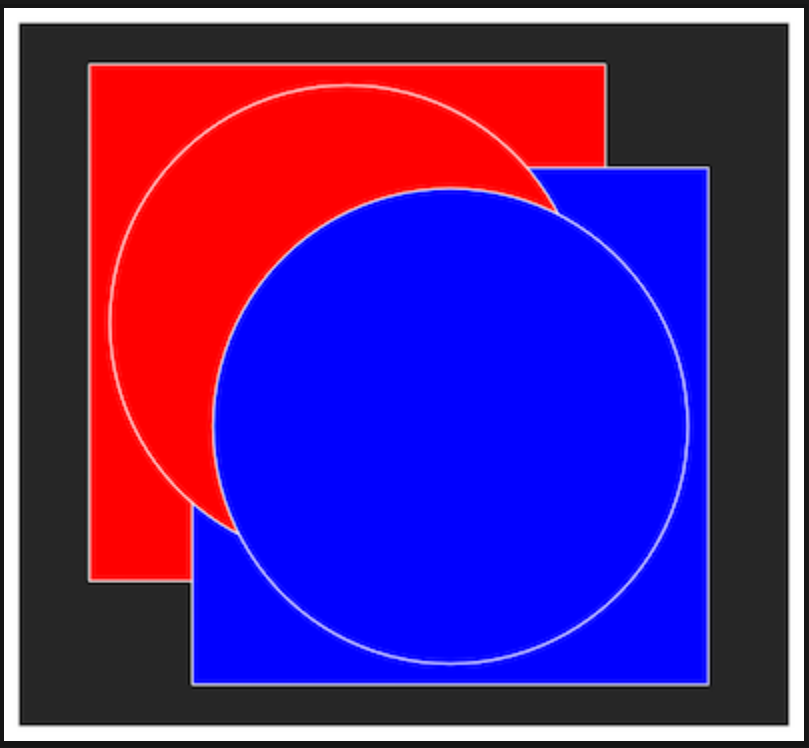
Because a child node's zPosition is added to its parent's zPosition, you can interleave child nodes from different parent nodes. Listing 1 shows code that creates two square parent nodes, each with a circular child node. The first node has a zPosition of 10 and its child node has a zPosition of 10. The second node has a zPosition of 15 and its child node also has a zPosition of 10. Once added to the scene, the child of the first node has an effective global zPosition of 20 and the child of the second node has an effective global zPosition of 25, giving an interleaving effect.

LISTING 1 INTERLEAVING COMPOSITE NODES

```
func addNodesTo(scene: SKScene,
                color: SKColor, position: CGPoint,
                parentZ: CGFloat, childZ: CGFloat) {

    let diameter: CGFloat = 250
    let rect = CGRect(origin: position,
                      size: CGSize(width: diameter, height: diameter))

    let parentNode = SKShapeNode(rect: rect)
    let childNode = SKShapeNode(ellipseIn: rect.insetBy(dx: 10, dy: 10))

    [(parentNode, scene, parentZ), (childNode, parentNode, childZ)].forEach {
        node, parent, zPosition in

        node.fillColor = color
        node.strokeColor = .white
        node.zPosition = zPosition

        parent.addChild(node)
    }
}


addNodesTo(scene: scene,
           color: .red,
           position: CGPoint(x: 300, y: 300),
           parentZ: 10,
           childZ: 10)

addNodesTo(scene: scene,
           color: .blue,
           position: CGPoint(x: 350, y: 250),
           parentZ: 15,
           childZ: 10)
```

Figure 3 shows the resulting scene.

FIGURE 3 INTERLEAVED NODE HIERARCHIES USING ZPOSITION



Because zPosition is additive across parent-child node relationships, you can use both tree order and zPositions to determine your scene's rendering order. When rendering a complex scene, you should disable the sorting behavior and use the zPositions of nodes to create a deterministic scene order.

> **Important:** The SKCropNode and SKEffectNode node classes alter the scene rendering behavior. The children of these nodes are rendered independently as a separate node tree, and the results are rendered into the tree that contains the crop or effect node. For more information, see Working with Other Node Types.