Resizing a Sprite in Nine Parts

Scale a sprite using nine-part algorithm.

Overview

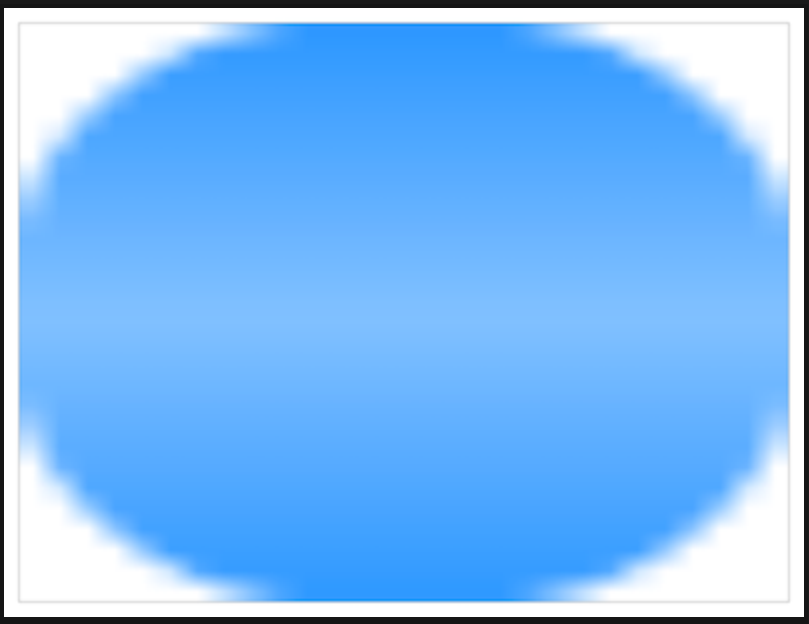
The size of the sprite node's `frame` property is determined by the values of these properties:

> >> The sprite node's size property holds its base (unscaled) size. When a sprite is initialized using `init(imageNamed:)`, the value of this property is initialized to be equal to the size of the supplied image.
>
> >> The base size is then scaled by the sprite's `xScale` and `yScale` properties inherited from the `SKNode`class.

For example, if the sprite node's base size is 32 x 32 pixels and it has an `xScale` value of 1.0 and a `yScale` value of 2.0, the size of its frame is 32 x 64 pixels.

Note: The scaling values of the sprite node's ancestors in the scene are also used to scale it. This changes the effective size of the sprite without changing its actual frame value. See A Node Applies Many of Its Properties to Its Descendants.

When a sprite node's frame is larger than its texture, the texture is stretched to cover its frame. Normally, the texture is stretched uniformly across the frame, as shown in the following figure.



However, sometimes you want to use sprite nodes to build user interface elements, such as buttons or health indicators. Often, these elements contain fixed-size elements, such as end caps, that should not be stretched. In this case, use a portion of the texture without stretching, and then stretch the remaining part of the texture over the rest of the frame.

The sprite's centerRect property, which is specified in unit coordinates of the texture, controls the scaling behavior. The default value is a rectangle that covers the entire texture, which is why the entire texture is stretched across the frame. If you specify a rectangle that only covers a portion of the texture, you create a 3 x 3 grid. Each box in the grid has its own scaling behavior:

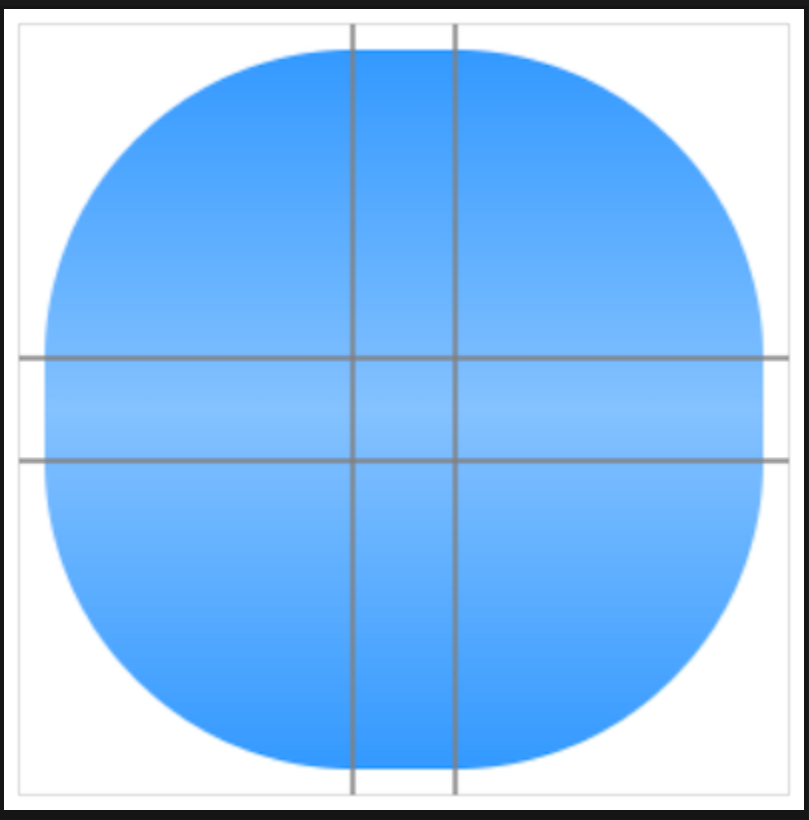
>> The portions of the texture in the four corners of the grid are drawn without any scaling.

>> The center of the grid is scaled in both dimensions.

>> The upper- and lower-middle parts are only scaled horizontally.

>> The left- and right-middle parts are only scaled vertically.

The following figure shows a close-up view of a texture you might use to draw a user interface button. The complete texture is 28 x 28 pixels. The corner pieces are each 12 x 12 pixels and the center is 4 x 4 pixels.



The following code shows how this button sprite would be initialized. The `centerRect` property is computed based on the design of the texture.

```
let button = SKSpriteNode(imageNamed: "stretchable_button.png")
button.centerRect = CGRect(x: 12.0/28.0,
                           y: 12.0/28.0,
                           width: 4.0/28.0,
                           height: 4.0/28.0)
```

The following figure shows that the corners remain the same, even when the button is drawn at different sizes.