Instance Method

aspectRatio(_:contentMode:)

Constrains this view's dimensions to the specified aspect ratio.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```swift
nonisolated
func aspectRatio(
    _ aspectRatio: CGFloat? = nil,
    contentMode: ContentMode
) -> some View
```

Show all declarations ⊕

Parameters

`aspectRatio`
The ratio of width to height to use for the resulting view. Use `nil` to maintain the current aspect ratio in the resulting view.

`contentMode`
A flag that indicates whether this view fits or fills the parent context.

Return Value

A view that constrains this view's dimensions to the aspect ratio of the given size using `contentMode` as its scaling algorithm.

Mentioned in

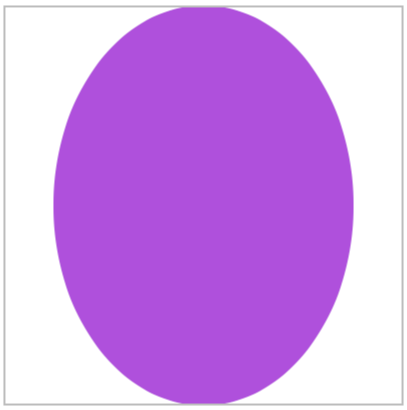
Fitting images into available space

Discussion

Use `aspectRatio(_:contentMode:)` to constrain a view's dimensions to an aspect ratio specified by a `CGFloat` using the specified content mode.

If this view is resizable, the resulting view will have `aspectRatio` as its aspect ratio. In this example, the purple ellipse has a 3:4 width-to-height ratio, and scales to fit its frame:

```
Ellipse()
    .fill(Color.purple)
    .aspectRatio(0.75, contentMode: .fit)
    .frame(width: 200, height: 200)
    .border(Color(white: 0.75))
```



See Also

Scaling, rotating, or transforming a view

`func scaledToFill() -> some View`

Scales this view to fill its parent.

`func scaledToFit() -> some View`

Scales this view to fit its parent.

`func scaleEffect(_:anchor:)`

Scales this view's rendered output by the given amount in both the horizontal and vertical directions, relative to an anchor point.

`func scaleEffect(x: CGFloat, y: CGFloat, anchor: UnitPoint) -> some View`

Scales this view’s rendered output by the given horizontal and vertical amounts, relative to an anchor point.

```
func scaleEffect(x: CGFloat, y: CGFloat, z: CGFloat, anchor: Unit
Point3D) -> some View
```

Scales this view by the specified horizontal, vertical, and depth factors, relative to an anchor point.

```
func rotationEffect(Angle, anchor: UnitPoint) -> some View
```

Rotates a view’s rendered output in two dimensions around the specified point.

```
func rotation3DEffect(Angle, axis: (x: CGFloat, y: CGFloat, z: CGFloat
), anchor: UnitPoint, anchorZ: CGFloat, perspective: CGFloat) -> some
View
```

Renders a view’s content as if it’s rotated in three dimensions around the specified axis.

```
func perspectiveRotationEffect(Angle, axis: (x: CGFloat, y: CGFloat, z:
CGFloat), anchor: UnitPoint, anchorZ: CGFloat, perspective: CGFloat) ->
some View
```

Renders a view’s content as if it’s rotated in three dimensions around the specified axis.

```
func rotation3DEffect(Rotation3D, anchor: UnitPoint3D) -> some View
```

Rotates the view’s content by the specified 3D rotation value.

```
func rotation3DEffect(_:axis:anchor:)
```

Rotates the view’s content by an angle about an axis that you specify as a tuple of elements.

```
func transformEffect(CGAffineTransform) -> some View
```

Applies an affine transformation to this view’s rendered output.

```
func transform3DEffect(AffineTransform3D) -> some View
```

Applies a 3D transformation to this view’s rendered output.

```
func projectionEffect(ProjectionTransform) -> some View
```

Applies a projection transformation to this view’s rendered output.

```
struct ProjectionTransform
```

```
enum ContentMode
```

Constants that define how a view’s content fills the available space.