Sample Code

Changing the appearance of selected and highlighted cells

Provide visual feedback to the user about the state of a cell and the transition between states.

Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 11.0+

Overview

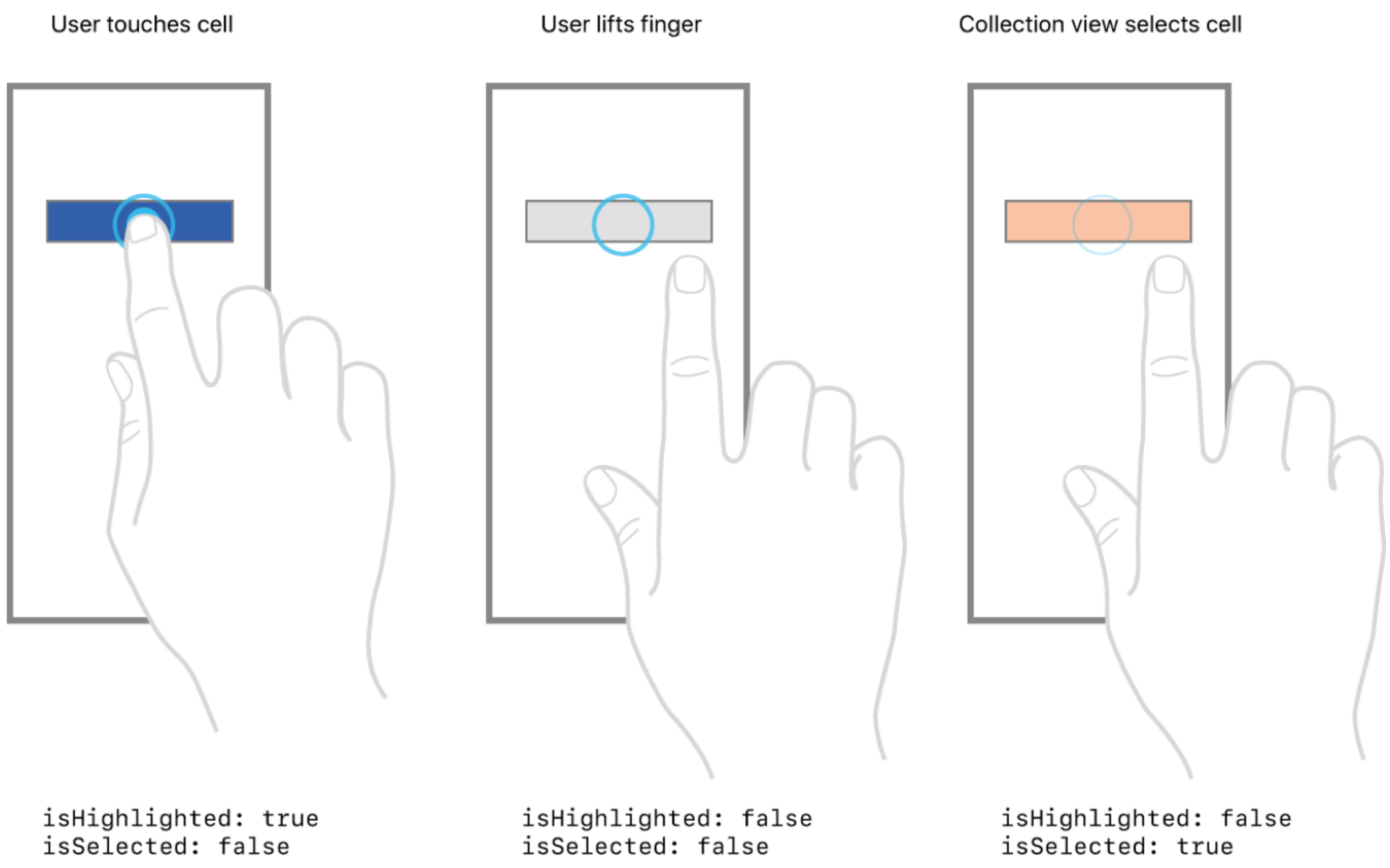
This sample app shows how to change the appearance of collection view cells as they transition between unselected, highlighted, and selected states. As a user taps a cell, the app determines the state of the cell and changes the cell's appearance to indicate the transition between states.

The collection view in this sample supports single-item selection, which is the default for collection views. You can also configure collection views to support multiple-item selection, or disable selection altogether.

Determine the state of a cell

The collection view in this sample determines the state of a cell by detecting the taps inside its bounds. It then sets the `isSelected` and `isHighlighted` properties of the corresponding cell to indicate the current state. The collection view provides this behavior because its `allowsSelection` property is set to `true`.

When touching an unselected cell, the initial touch-down event causes the collection view to change the `isHighlighted` property of the cell to `true`. The final touch-up event causes the highlighted state to return to `false`. If the touch-up event occurs inside the cell, the collection view sets the cell’s isSelected property to `true`; otherwise, the property value remains unchanged.



Change the cell’s visual appearance

The cell’s backgroundView property references the view to display as the background of the cell when it loads for the first time and when it isn’t highlighted or selected. When the cell’s state changes to highlighted or selected, the collection view modifies the properties of a cell to indicate the new state. It doesn’t, however, automatically change the visual appearance of the cell. That is, unless you set the cell’s selectedBackgroundView property to a view.

Setting the selectedBackgroundView to a view causes the collection view to replace the default background with the selected background when highlighting or selecting a cell. Your app doesn’t need to do anything else. The collection view changes the cell’s visual appearance automatically as the cell’s state changes. For example, the sample app uses the selectedBackgroundView property to change the cell’s background color from red to blue when selecting the cell.

```
override func awakeFromNib() {
    super.awakeFromNib()

    let redView = UIView(frame: bounds)
    redView.backgroundColor = #colorLiteral(red: 1, green: 0, blue: 0, alpha: 1)
```

```swift
    self.backgroundView = redView

    let blueView = UIView(frame: bounds)
    blueView.backgroundColor = #colorLiteral(red: 0, green: 0, blue: 1, alpha: 1)
    self.selectedBackgroundView = blueView
}
```

Provide additional visual indication of state changes

Providing a selected background view in a cell is an easy way to have the collection view change the cell's appearance based on its state, but you can do more than just changing the background. You can, for example, display a checkmark icon in selected cells or distinguish between highlighted and selected states visually.

The collection view's `delegate` has methods that provide you many opportunities to tweak the selection and highlighting appearance of your collection view. For example, if you prefer to draw the selection state of the cell yourself, you can leave the `selectedBackgroundView` property set to `nil`. Then apply any visual changes to the cell in the `collectionView(_:didSelectItemAt:)` delegate method. The sample app uses this method, along with the selected background, to show a star in the selected cell. The app removes the star in the `collectionView(_:didDeselectItemAt:)` delegate method.

```swift
func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: I
    if let cell = collectionView.cellForItem(at: indexPath) as? CustomCollectionView
        cell.showIcon()
    }
}

func collectionView(_ collectionView: UICollectionView, didDeselectItemAt indexPath:
    if let cell = collectionView.cellForItem(at: indexPath) as? CustomCollectionView
        cell.hideIcon()
    }
}
```

If you prefer to draw the highlight state, use the `collectionView(_:didHighlightItemAt:)` and `collectionView(_:didUnhighlightItemAt:)` delegate methods. The sample app uses these two methods to display a different shade of red as the highlighted background color. Because cells in this app use the blue view for their `selectedBackgroundView`, the delegate must apply its changes to the content view of the cells to ensure the changes are visible.

```
func collectionView(_ collectionView: UICollectionView, didHighlightItemAt indexPath
    if let cell = collectionView.cellForItem(at: indexPath) {
        cell.contentView.backgroundColor = #colorLiteral(red: 1, green: 0.4932718873
    }
}

func collectionView(_ collectionView: UICollectionView, didUnhighlightItemAt indexPa
    if let cell = collectionView.cellForItem(at: indexPath) {
        cell.contentView.backgroundColor = nil
    }
}
```

See Also

Selection management

{} Selecting multiple items with a two-finger pan gesture

Accelerate user selection of multiple items using the multiselect gesture on table and collection views.