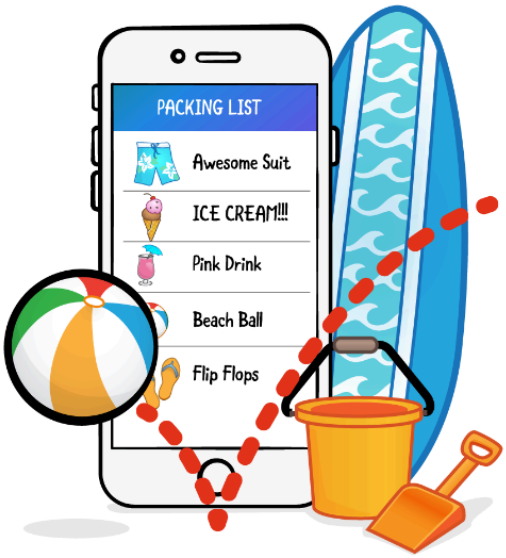
BEGINNING iOS ANIMATIONS



Hands-On Challenges

Beginning iOS Animations

Catie & Jessy Catterwaul

Notice of Rights

Notice of Liability

Trademarks

Table of Contents: Overview

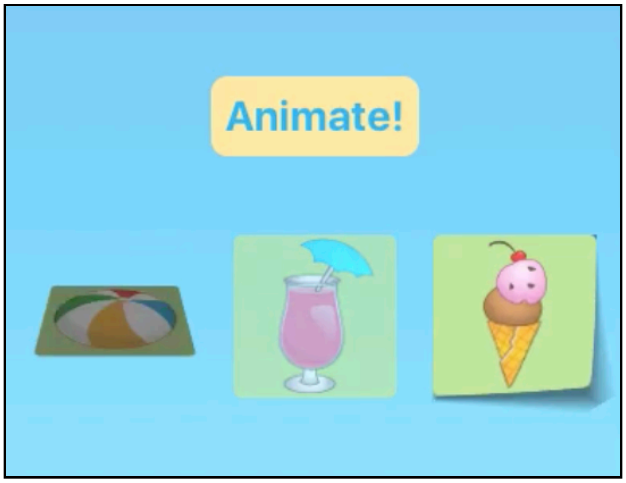
Table of Contents: Extended

Challenge 5: View Transitions

By Catie & Jessy Catterwaul

Part 1 - Experiment with View Transitions

Transitions.playground can be found in your downloaded materials and contains all of the instructions you need for this part of the challenge.



Part 2 - Finish up the Packing List project

In this challenge you are going to iterate over creating view transitions by creating one more animation of that type.

In the video you learned how to trigger view transitions by hiding and showing a view. This time you will do a similar effect by actually removing the view from the view hierarchy.

The key difference is that transitions triggered via removing views require a separate container view. Any views added to or removed from this container view will be animated with the transition you specify. In a transition triggered

by `.isHidden`, the view you are hiding or showing can act as its own container view.

Open ViewController.swift and scroll to `transitionCloseMenu()`. This method currently just calls `actionToggleMenu()` to close up the menu after a short delay. Here you are going to add a bit of code to trigger a transition animation just before the menu closes.

In the same method, after the call to delay, get hold of the menu and store a reference in a local constant to use as the container view:

```
      let titleBar = slider.superview!
```

Add the transition animation call, just like you did in the demo:

```
  UIView.transition(
    with: titleBar,
    duration: 0.5,
    options: [
      .curveEaseOut,
      .transitionFlipFromBottom
    ],
    animations: {

    },
    completion: {_ in

    }
  )
```

The container view for the transition is the menu and you are using a flip from bottom animation. So far so good - now you need also the code to trigger the transition inside the `animations` closure:

```
      self.slider.removeFromSuperview()
```

This will remove slider from the view hierarchy and since `titleBar` is `slider`'s parent view, that will trigger the transition animation.

Since you don't want to permanently remove the slider you can add it back to the view hierarchy once the animation is complete. Add inside `completion`:

```
      titleBar.addSubview(self.slider)
```

Run the app and open the menu. To test your new transition animation just select an item from the horizontal list and that will make it flip over and close the menu. Back flip for the win :]

Part 3 - One last thing...

Sometimes these view transitions, or any animation, may take longer than your

user would like. By default, a view that is mid-animation cannot be interacted with. However, another handy trick hidden in `options` is `.allowUserInteraction`. This will allow your users to interact with the views while they are animating.

Try it out in the animation of `actionToggleMenu`. Add `.allowUserInteraction` to the empty set:

```
  options: [.allowUserInteraction]
```

Use this option with care, because it can cause a performance hit. Views that should always be interactive, like buttons or scroll views, are a good use case.