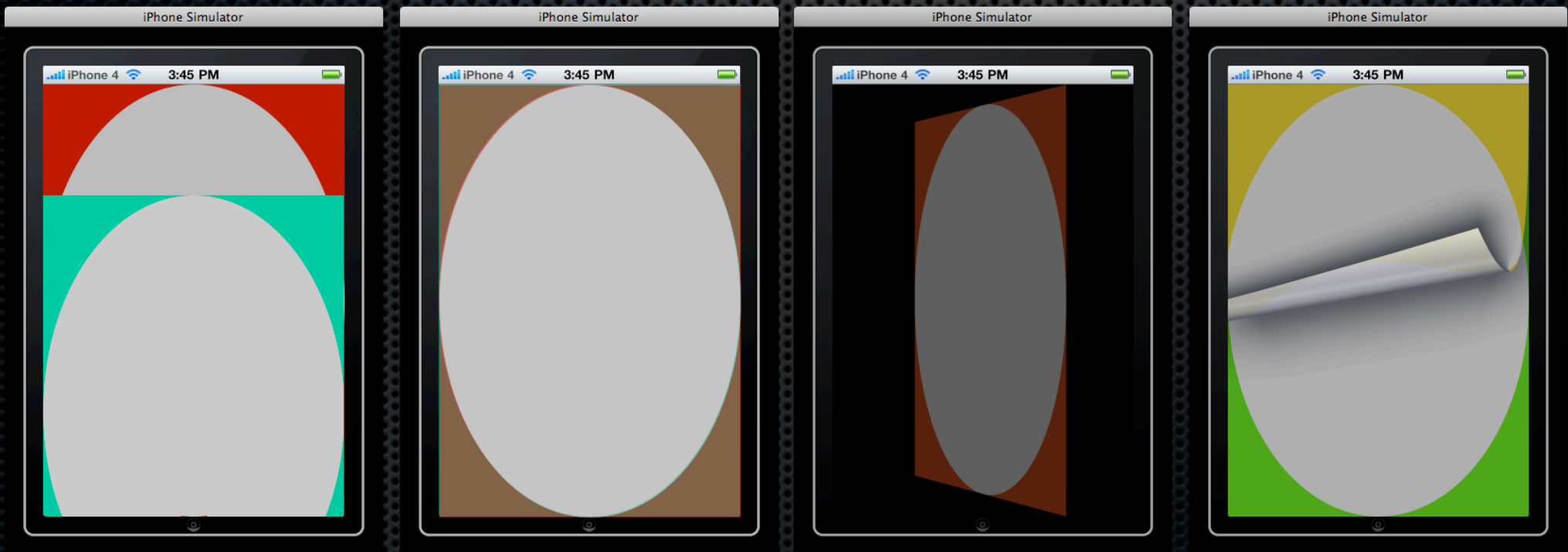


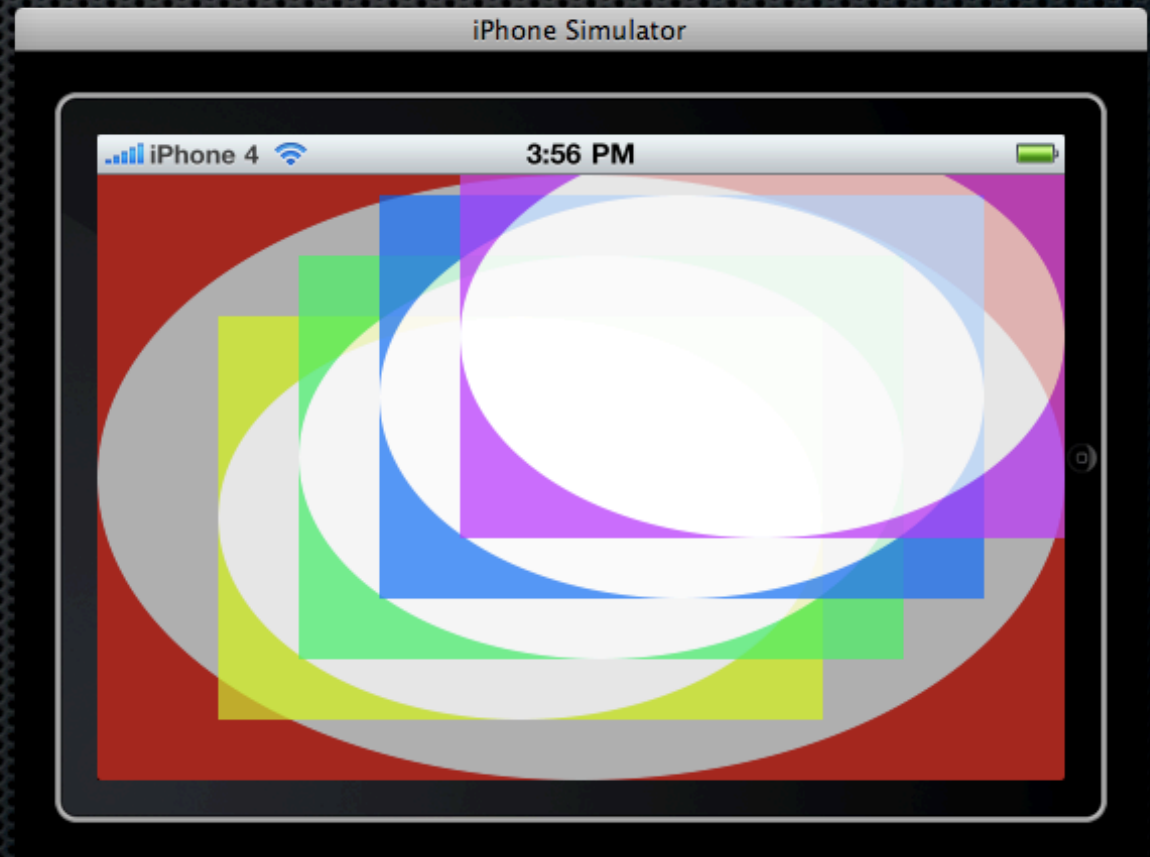
UofU iPhone Group

Container View Controllers

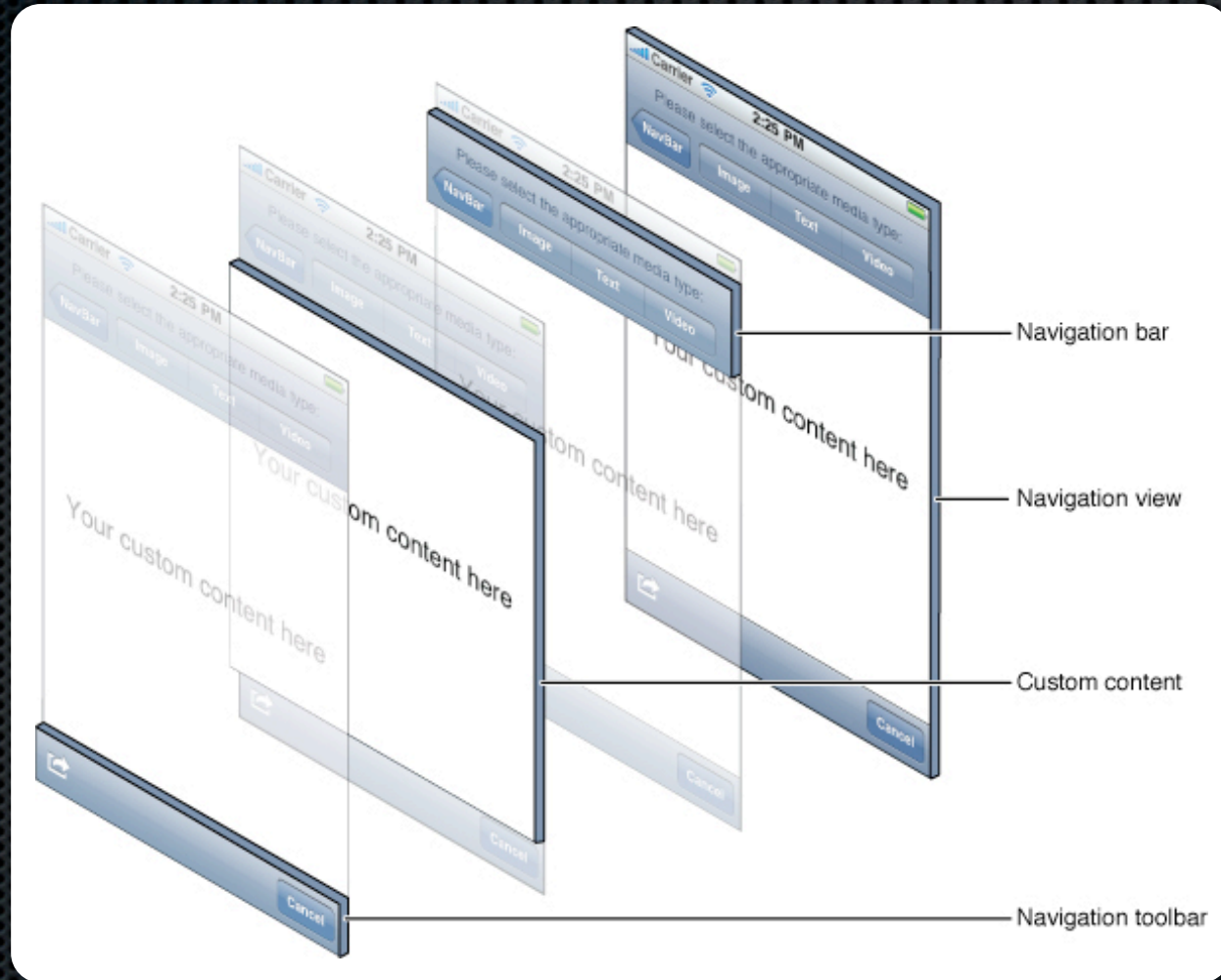
Simple Presentation



Simple Presentation Issues



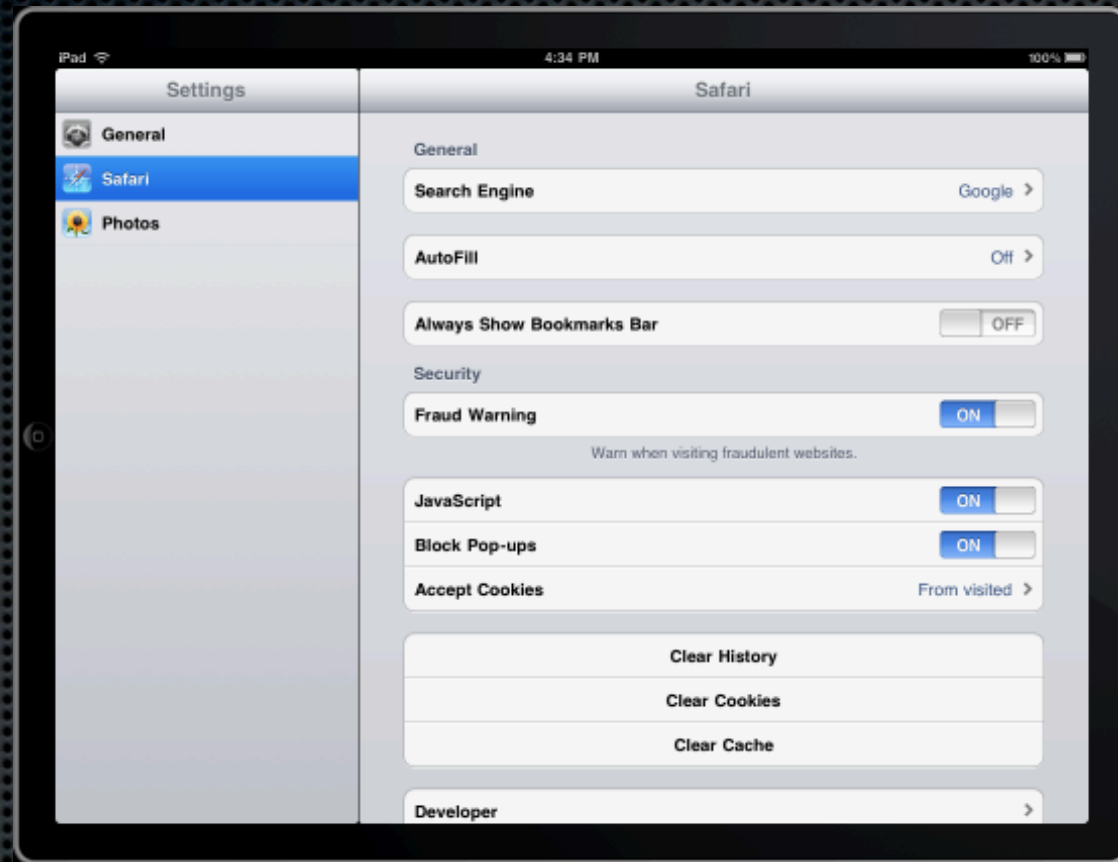
Navigation Controllers

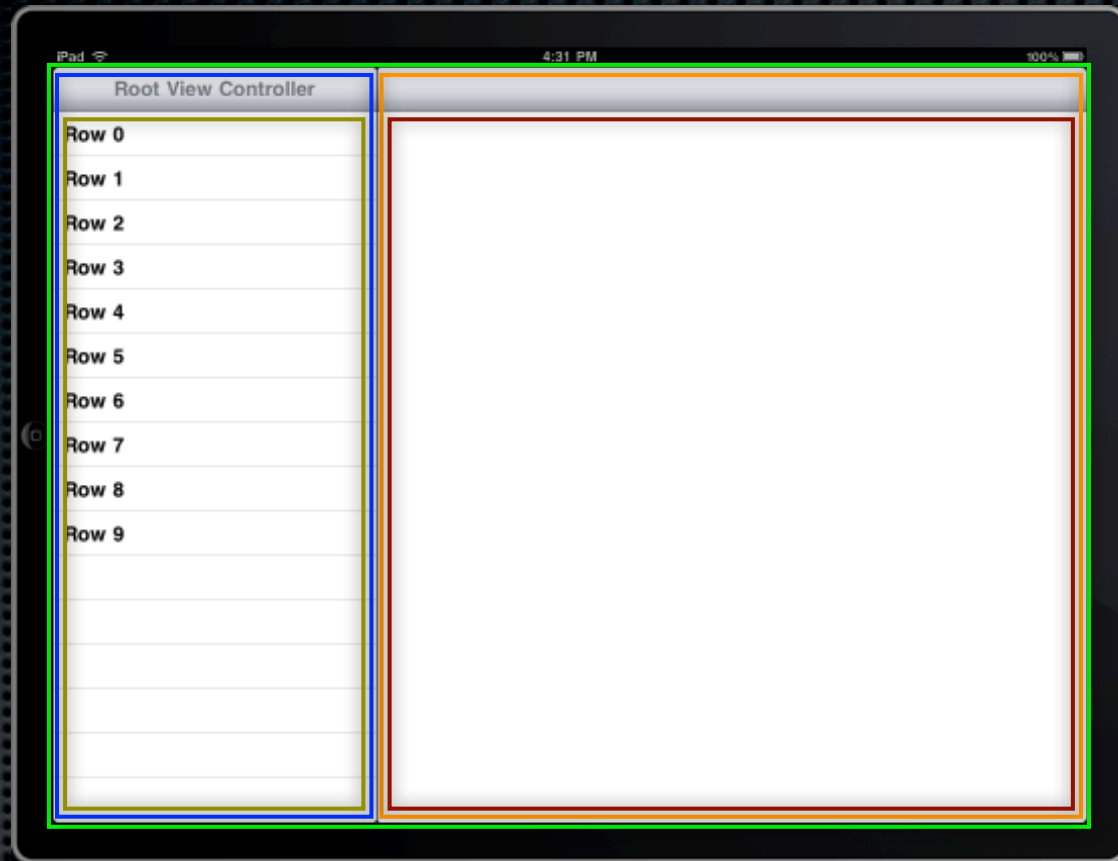


Tab Bar Controllers



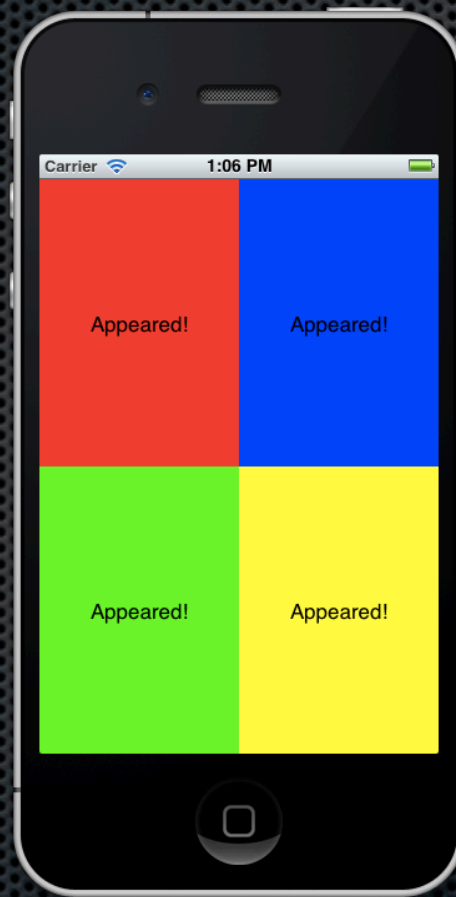
Split-View Controllers



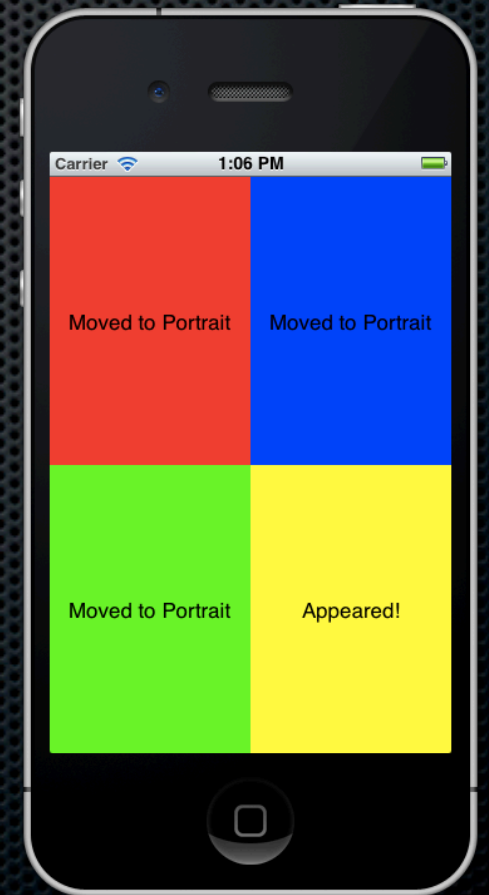
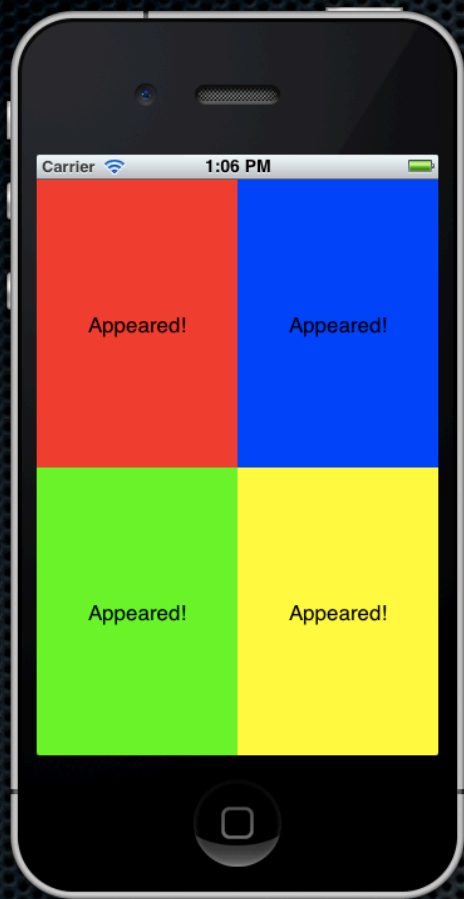


- Split-View Controller
- Master Navigation Controller
- Detail Navigation Controller
- Master View Controller
- Detail View Controller

Container View Controllers



Container View Controllers



Container View Controllers

```
viewWillAppear:  
viewDidAppear:  
viewWillDisappear:  
viewDidDisappear:  
willRotateToInterfaceOrientation:duration:  
willAnimateRotationToInterfaceOrientation:duration:  
didRotateFromInterfaceOrientation:
```

Container View Controllers

Implementing a Container View Controller

In iOS 5.0 and later, custom `UIViewController` subclasses can now act as container view controllers. The `UINavigationController` and `UITabBarController` classes are examples of container view controllers provided by UIKit. The idea behind a container view controller is that it manages the presentation of the content from its contained view controllers, also known as its child view controllers. The child content can be presented as-is or in conjunction with other other custom views managed by the container view controller.

To implement a container view controller, make a custom `UIViewController` subclass that calls the view containment methods as appropriate for your containment model:

```
addChildViewController:  
removeFromParentViewController  
transitionFromViewController:toViewController:duration:options:animations:completion:  
willMoveToParentViewController:  
didMoveToParentViewController:
```

Note: You are not required to override any methods in order to create a container view controller.

You may optionally override the `automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers` method.

For example, a navigation controller would call the `addChildViewController:` and `transitionFromViewController:toViewController:duration:options:animations:completion:` methods from its `pushViewController:animated:` method, and it would call the `removeFromParentViewController` and `transitionFromViewController:toViewController:duration:options:animations:completion:` method from its `popViewControllerAnimated:` method.

Likewise, a tab bar controller would call the `addChildViewController:` method once for each child view controller at startup, and it would call the `transitionFromViewController:toViewController:duration:options:animations:completion:` method when transitioning between tabs.

The `addChildViewController:` method calls the `willMoveToParentViewController:` method of the view controller to be added as a child before adding it, but it does not call the `didMoveToParentViewController:` method. The container view controller class must call the `didMoveToParentViewController:` of the child view controller after the transition to the new child is complete or, if there is no transition, immediately after calling the `addChildViewController:` method.

Likewise, it is the responsibility of the container view controller to call the `willMoveToParentViewController:` method before calling the `removeFromParentViewController` method. The `removeFromParentViewController` method calls the `didMoveToParentViewController:` method of the child view controller.