Last week

Last week

- C
- Objects
- Classes
- Protocols
- Blocks
- Memory management

Building your own app

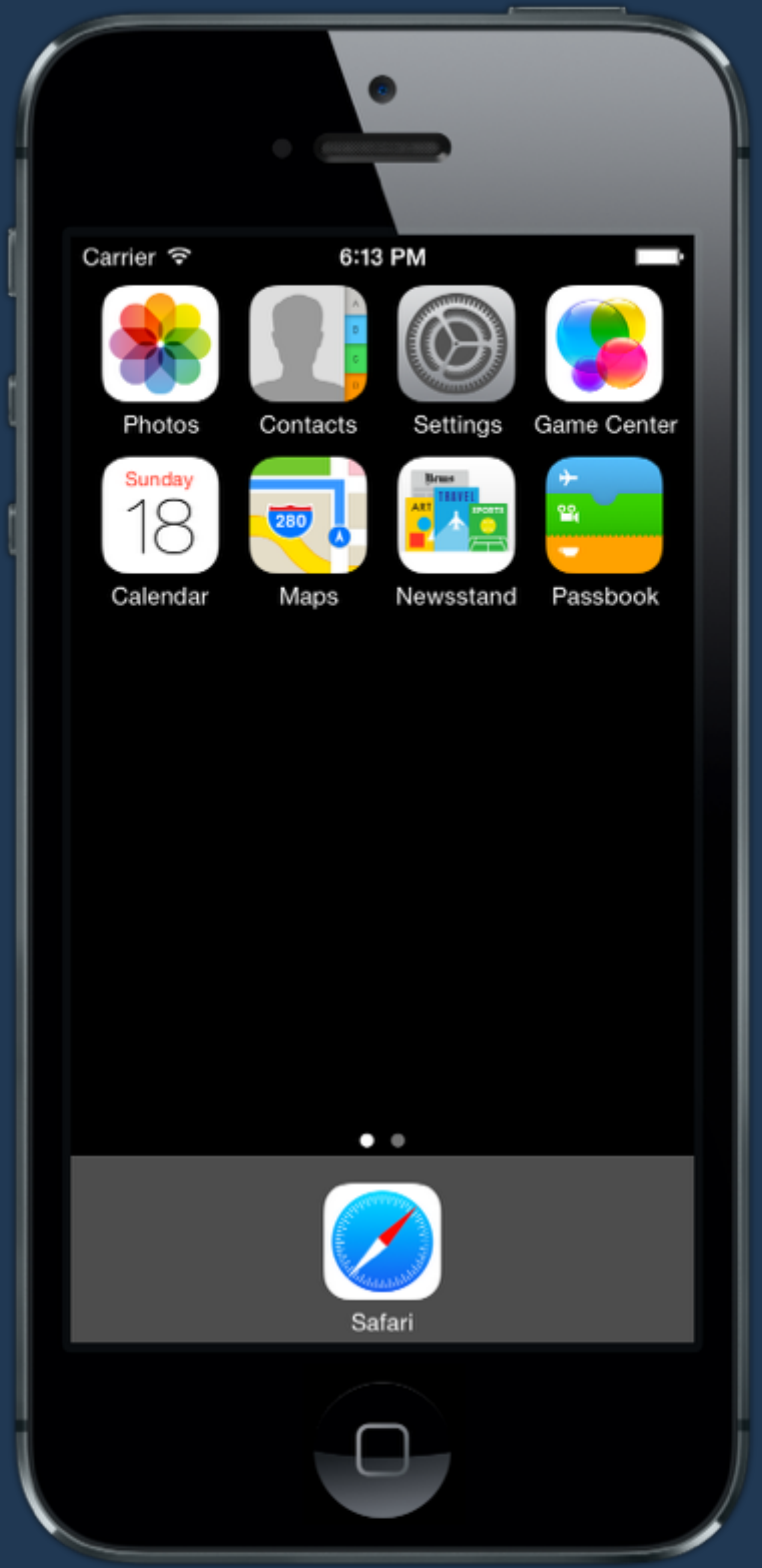
Smarticle

Smarticle

- List the most popular articles from New York Times API
- Add articles as favorites
- View details for a particular article
- Read articles

Apps

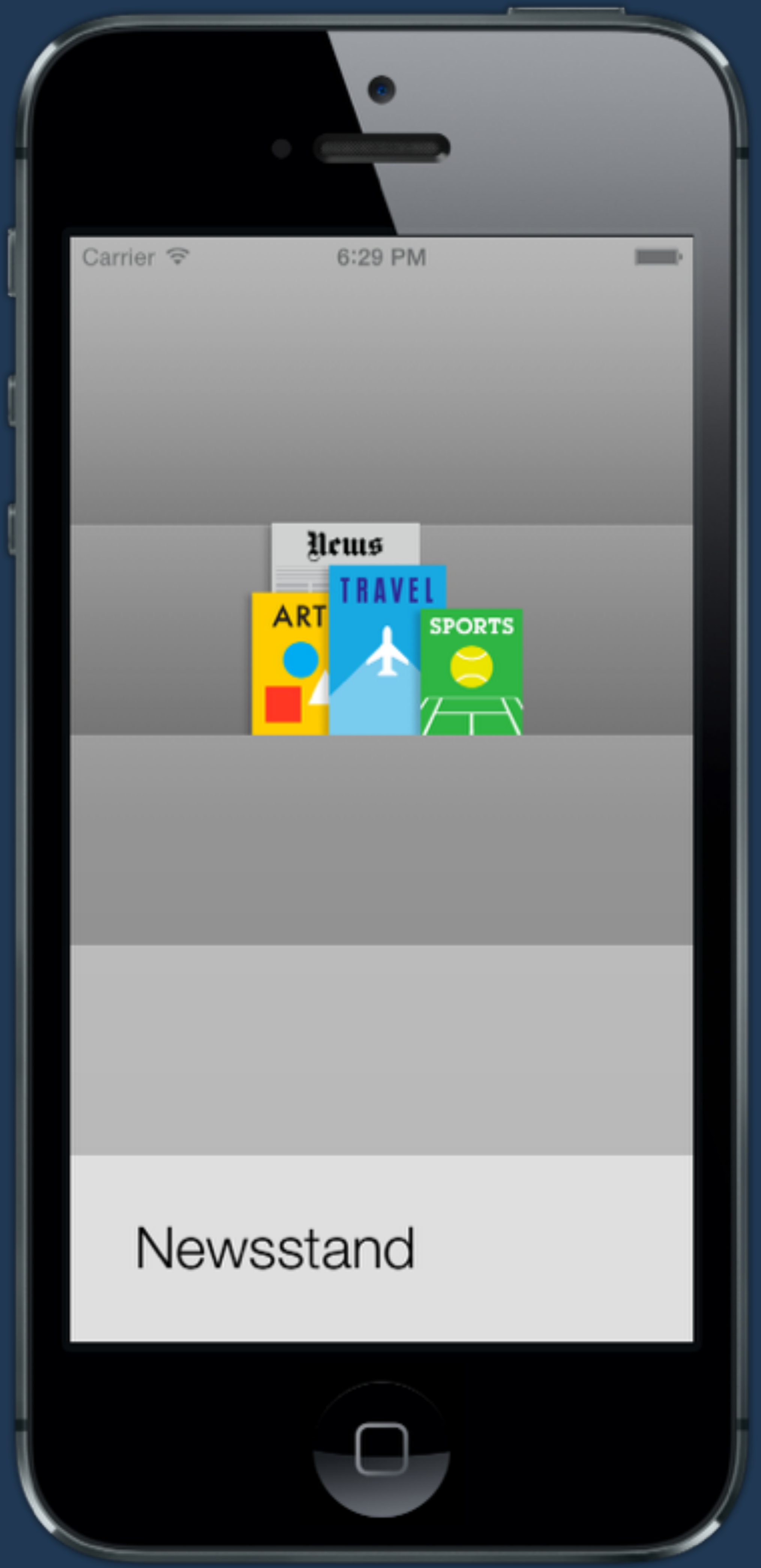
What is an app?



From the user's perspective

- A self-contained program
- Interactive display of information
- An excuse to complain on the App Store

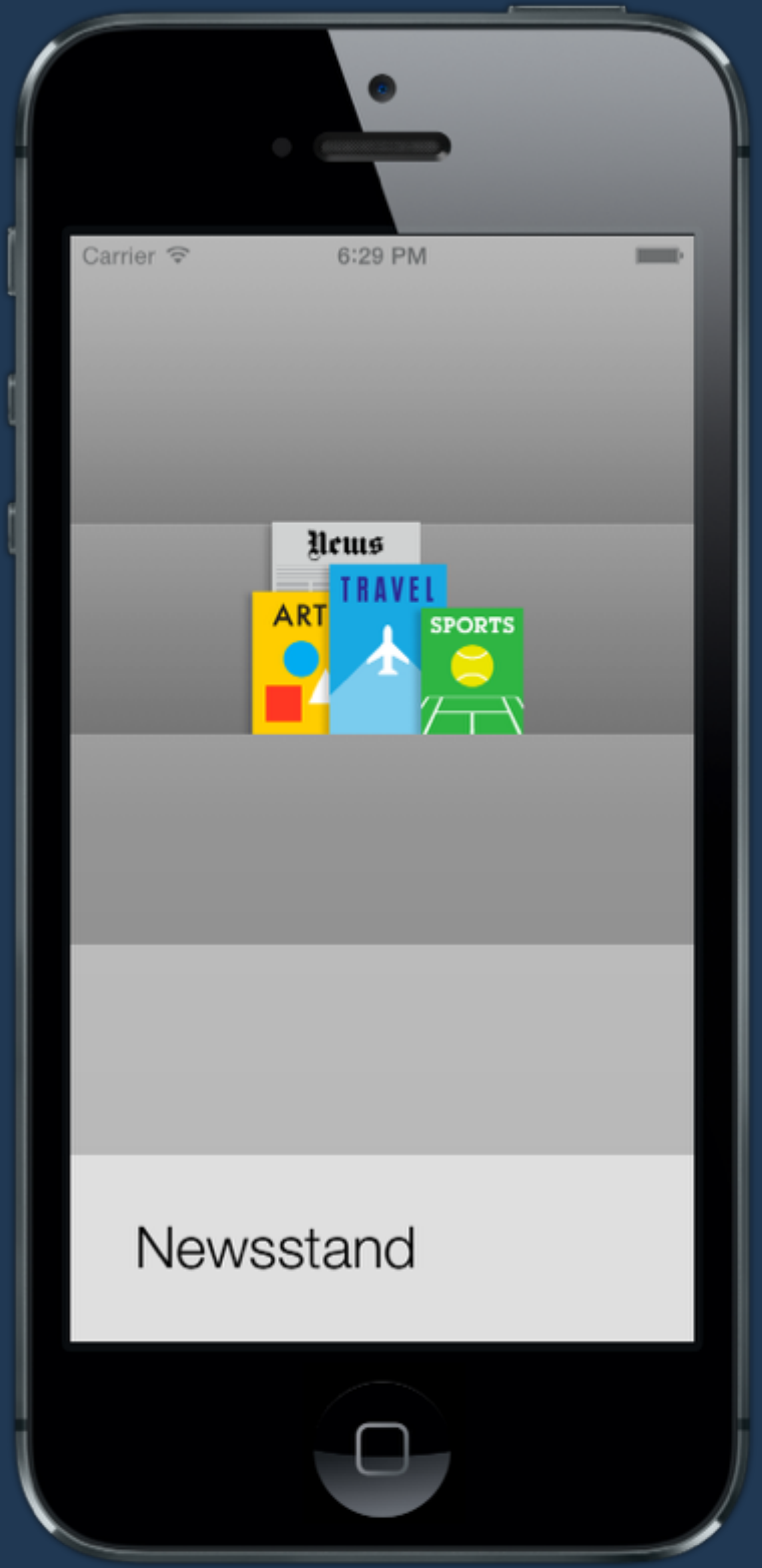
What is an app?



From iOS’ perspective

- Launched in a single process
- 4 GB virtual memory constrained by RAM
- Sandboxed for security

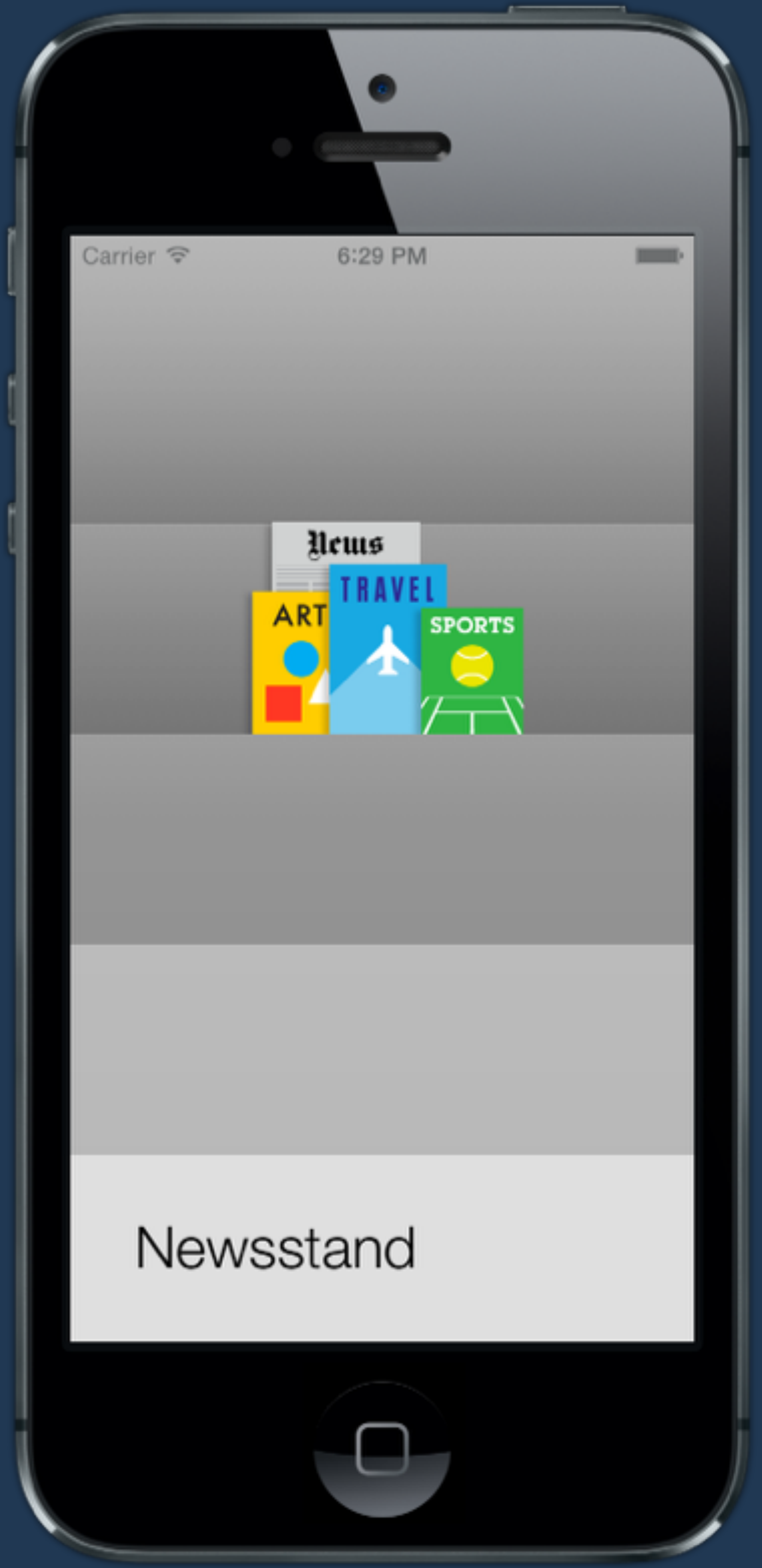
What is an app?



From your perspective

- Data
- Behavior

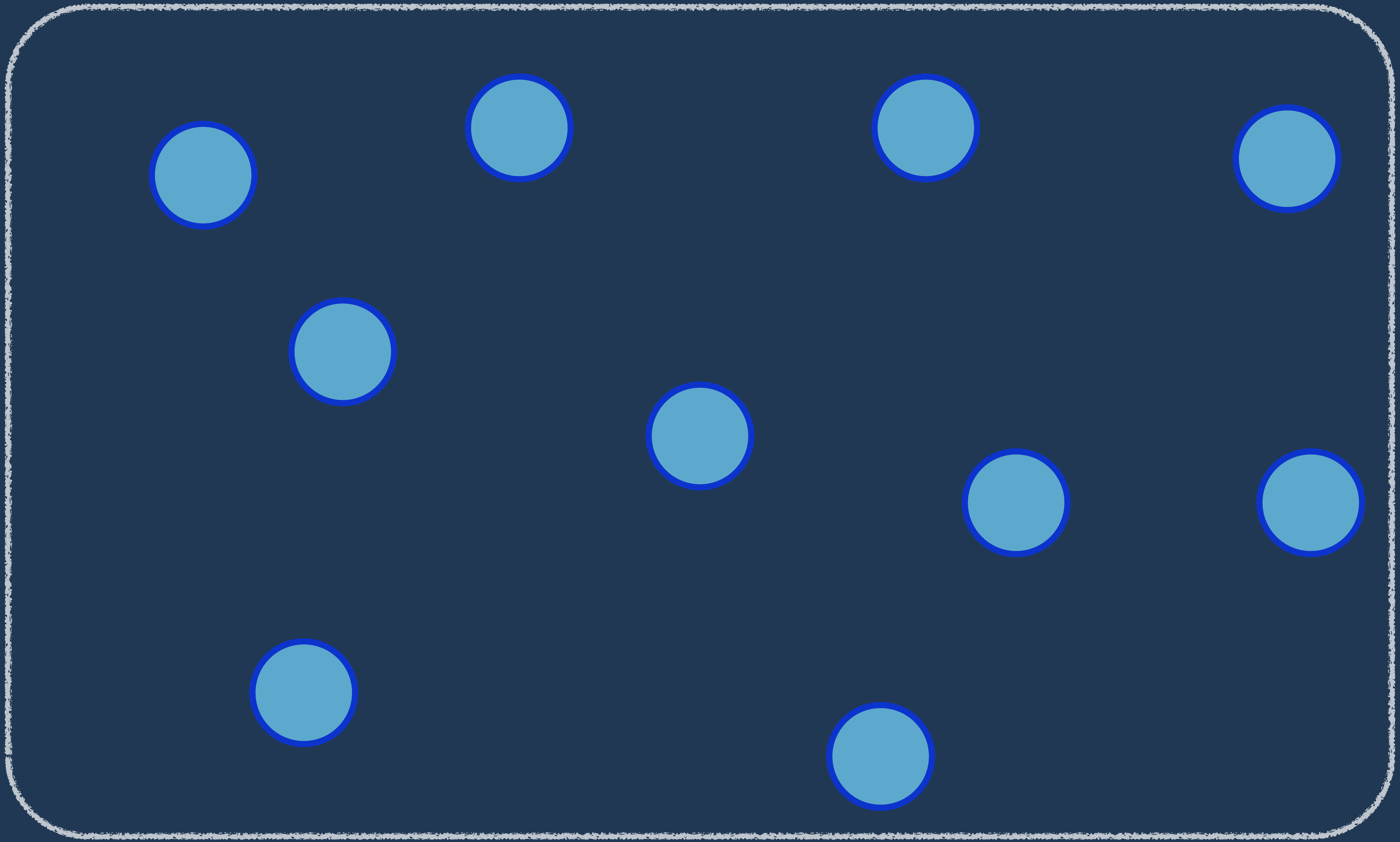
What is an app?



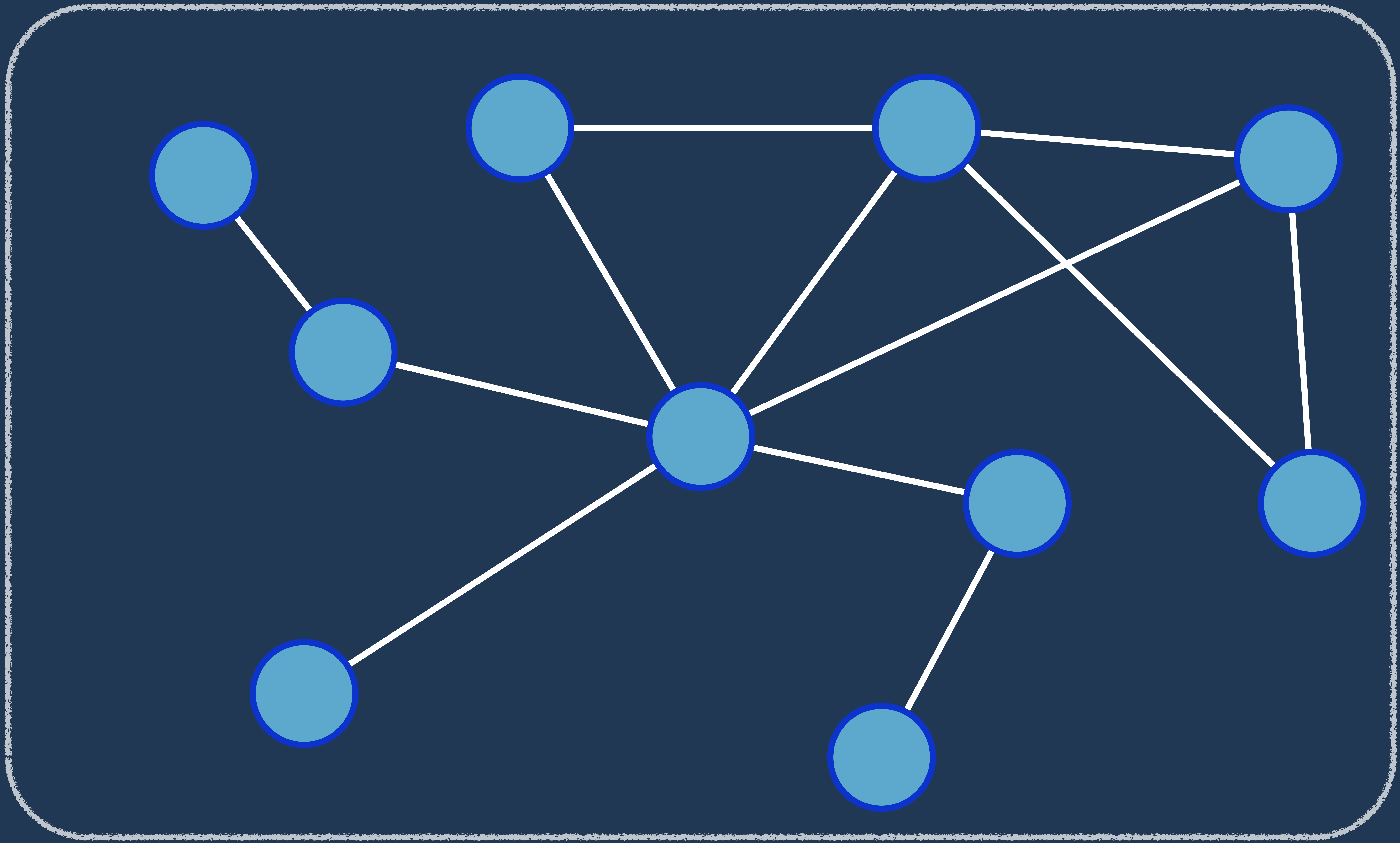
From your perspective

- Objects
- Interactions between objects

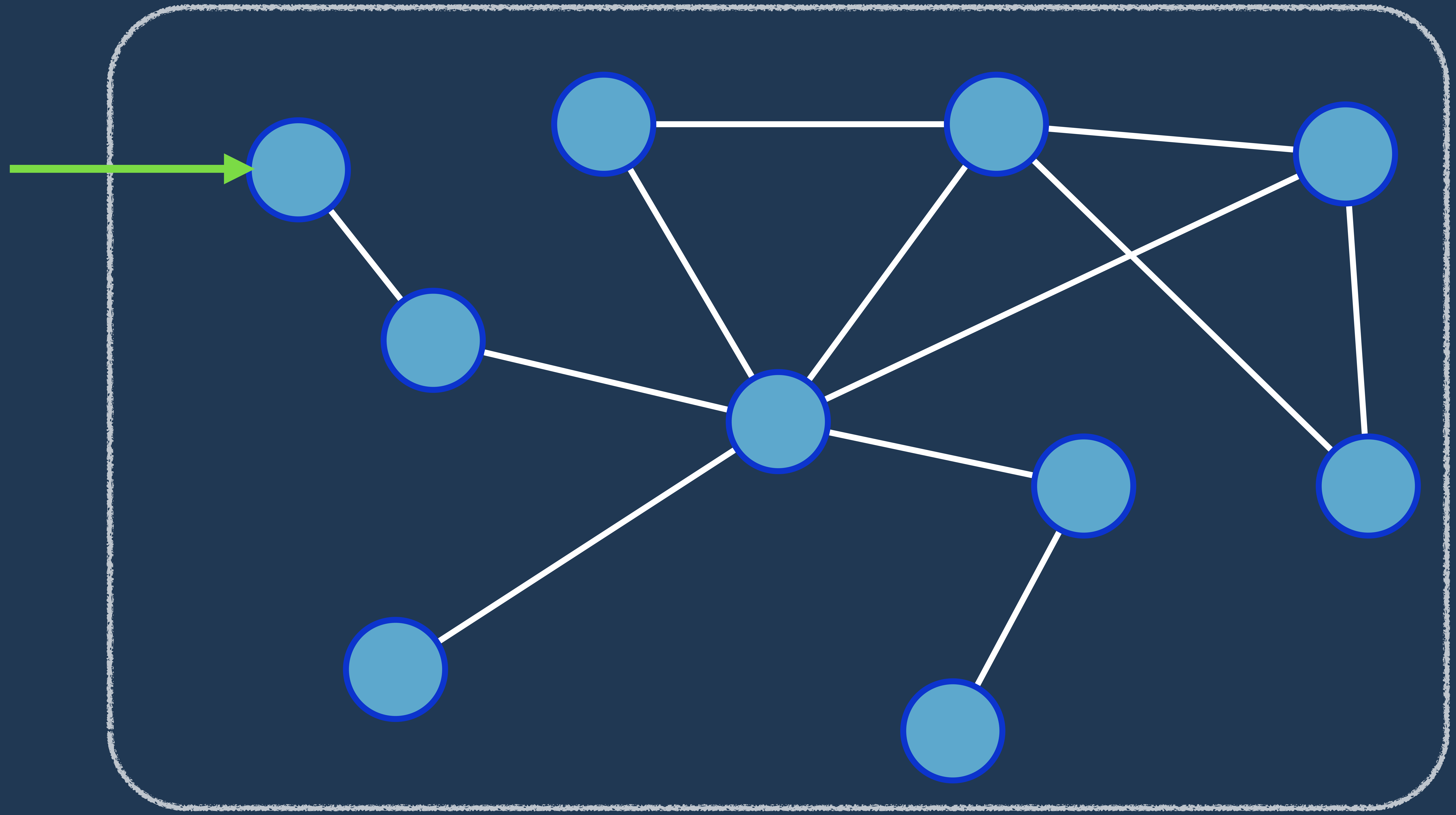
App



App



A starting point



Create a project for Smarticle

Lab 2.1

main.m

```
int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

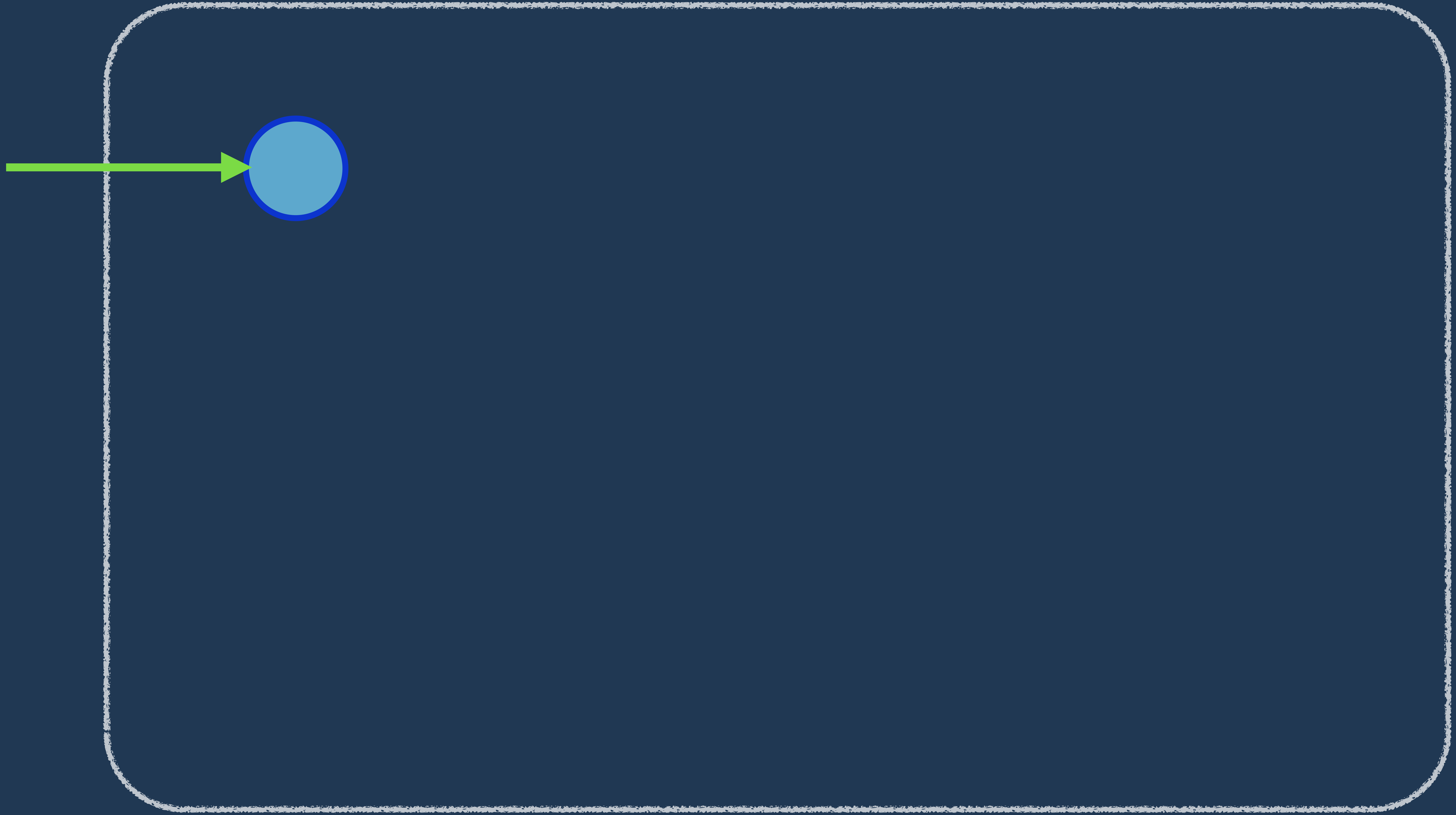
main.m

```
#import <UIKit/UIKit.h>

#import "AppDelegate.h"

int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

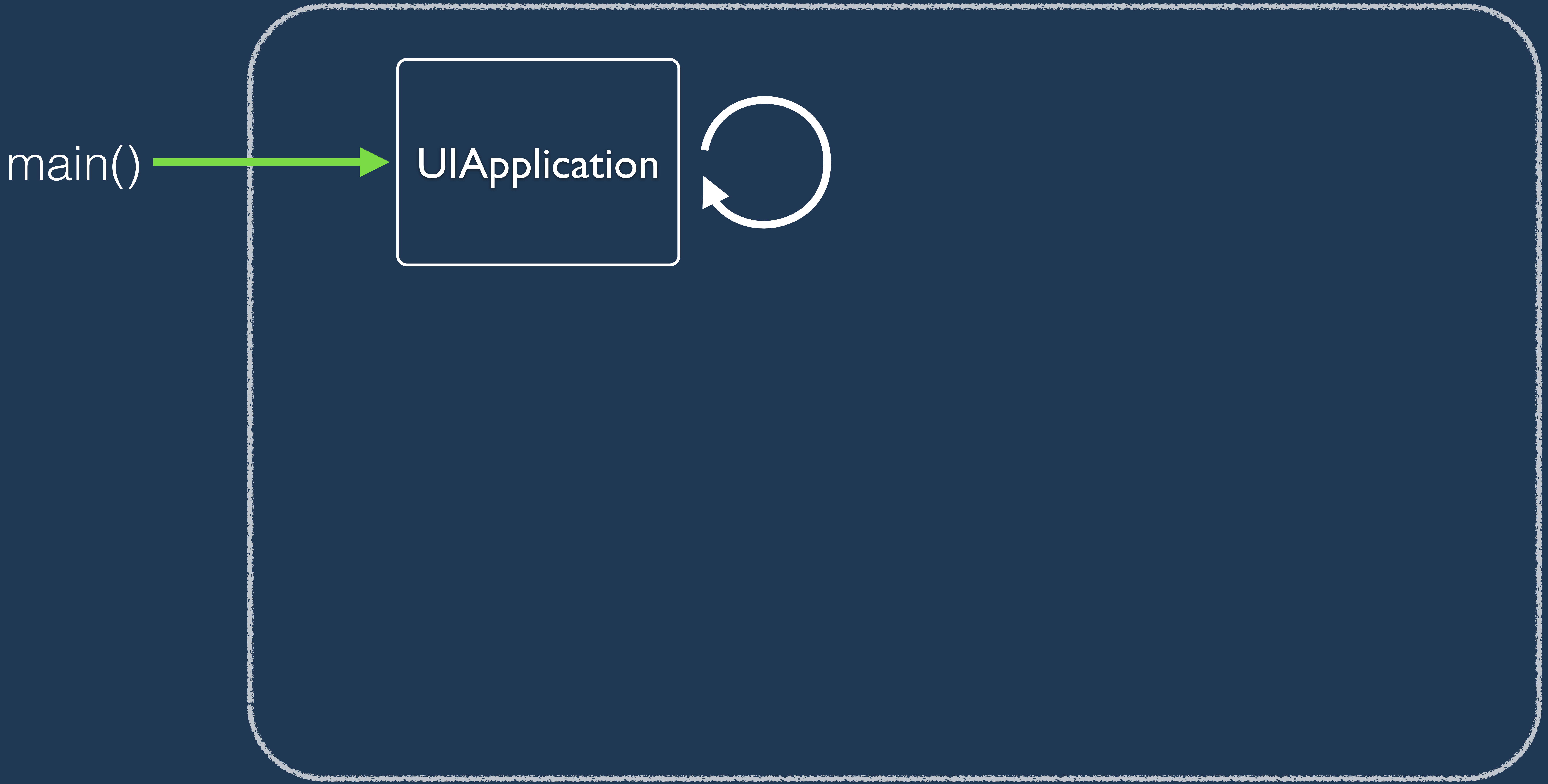
UIApplicationMain()

- Part of the UIKit framework
- Creates the UIApplication object, your app’s first object
- Starts an infinite loop, known as the run loop

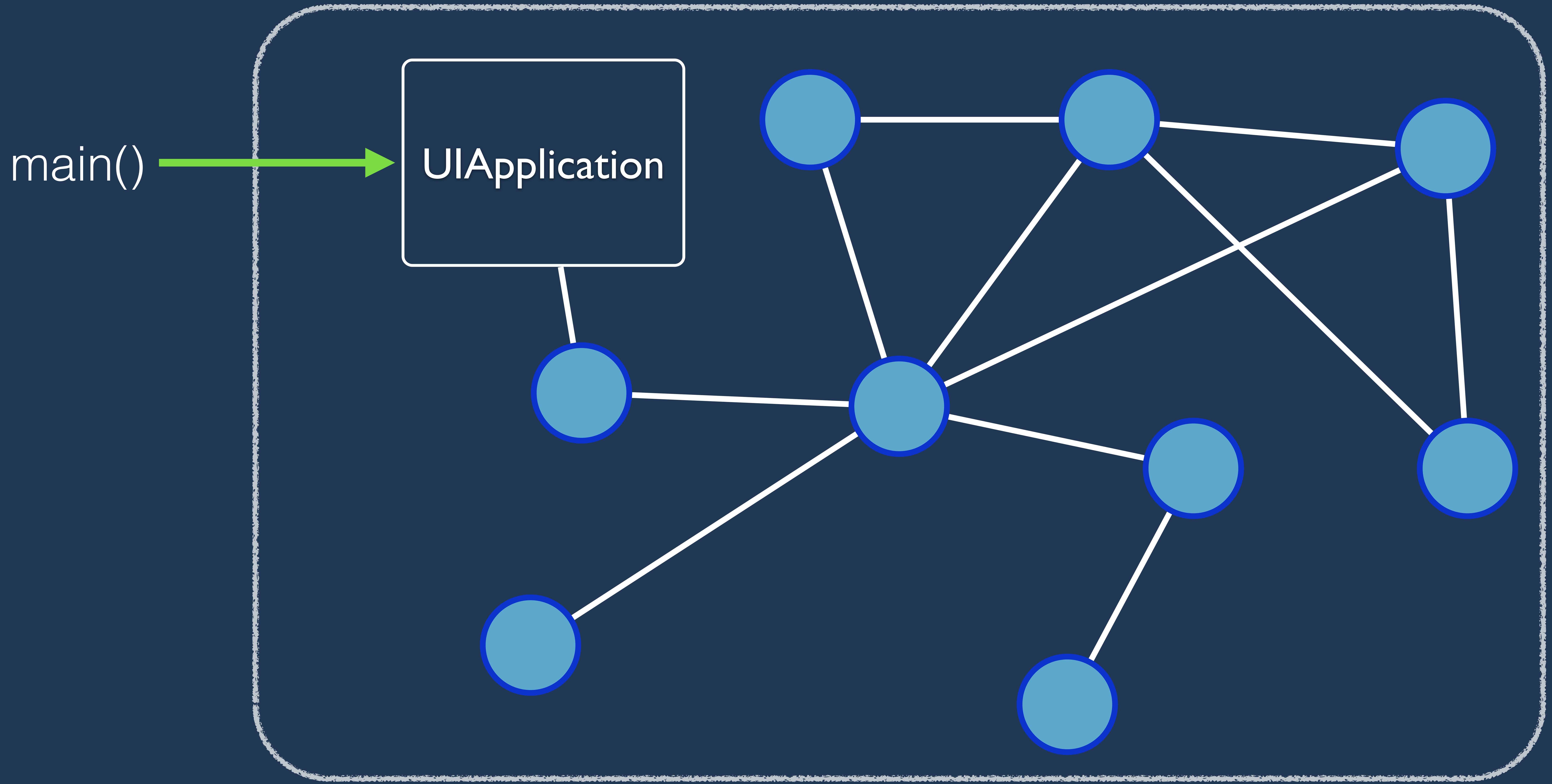
UIApplicationMain()



UIApplicationMain()



UIApplicationMain()



Design Patterns

Design Patterns

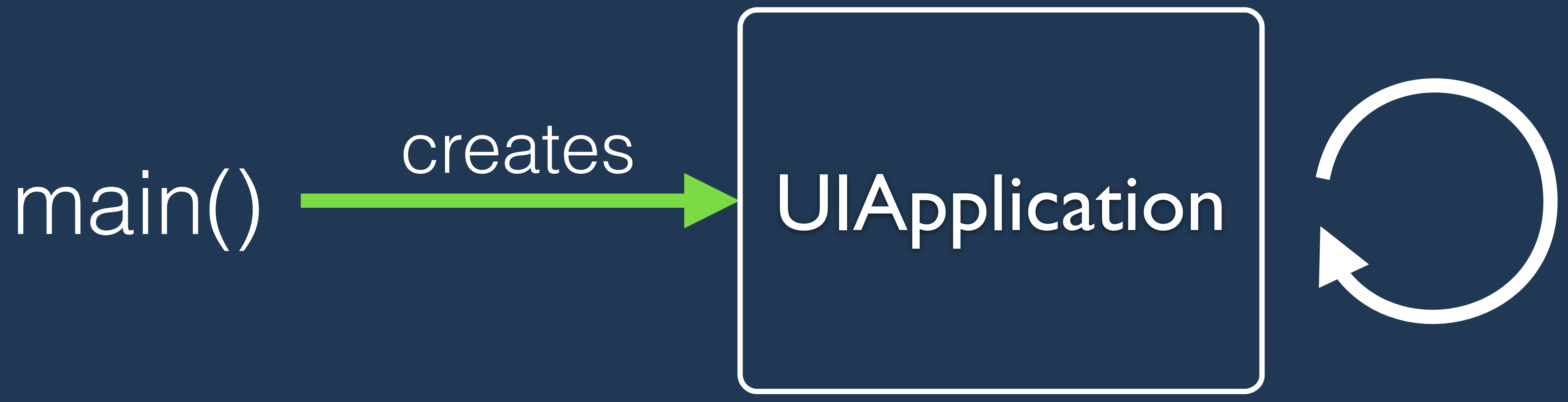
- Delegate
- Model View Controller
- Target-action

Delegate

Delegate

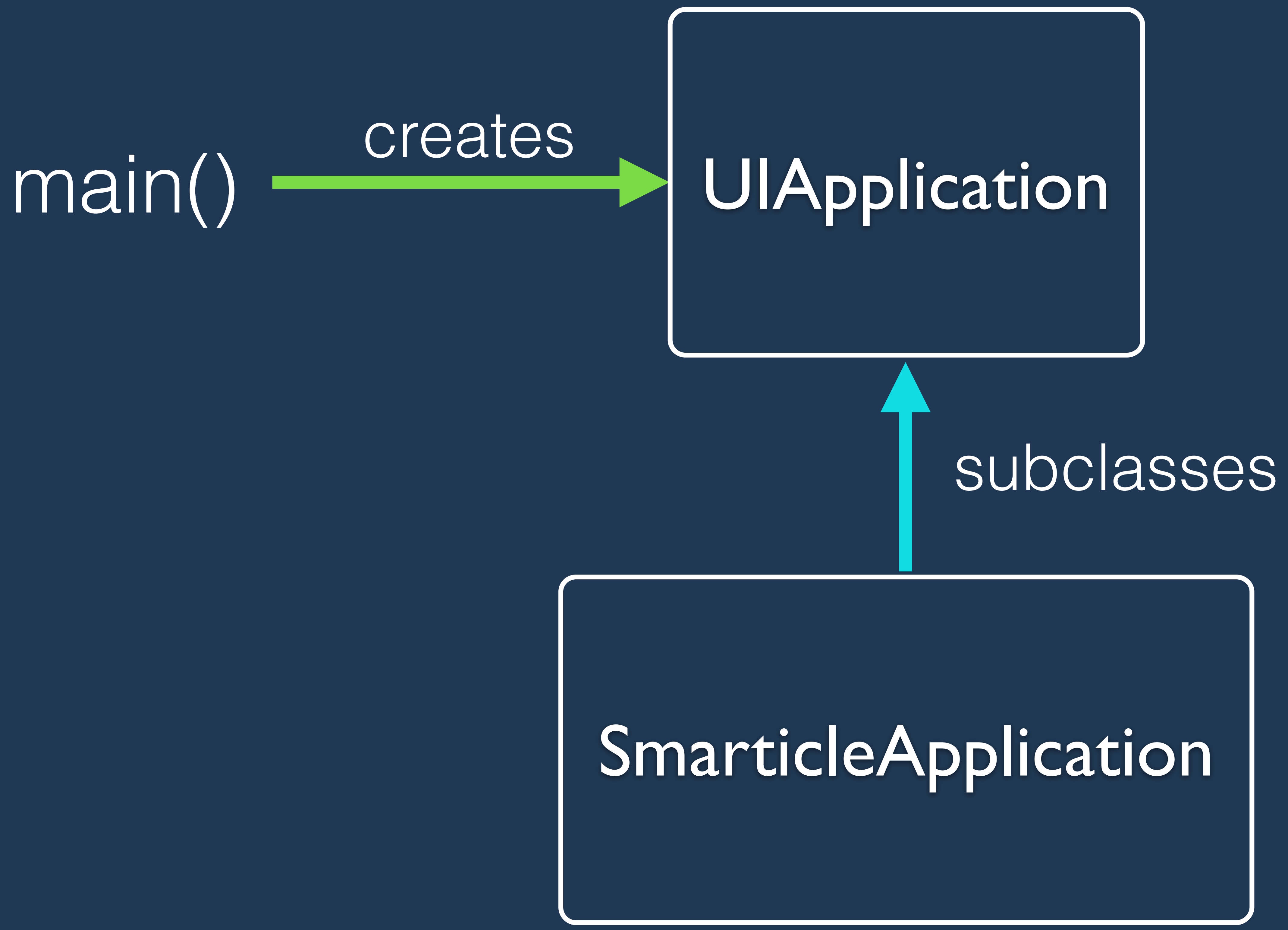
- Implemented as a protocol
- Transfers responsibility from one object to another
- Modularity, without subclassing

If we were Apple...

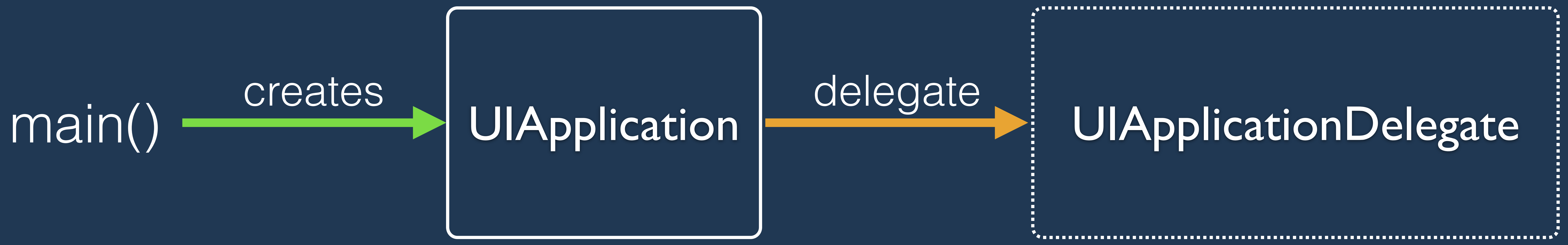


We’d need a way for app developers to start customizing their apps

A subclass approach



The App Delegate


main()

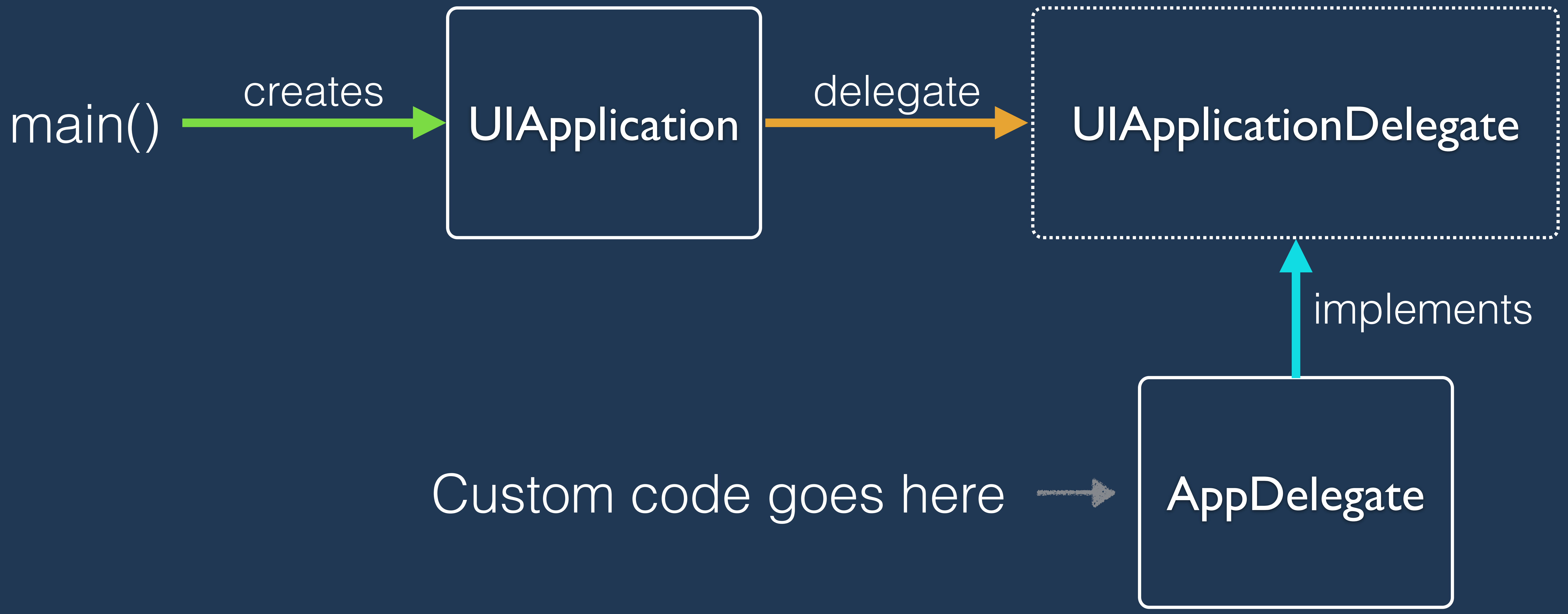
creates

UIApplication

delegate

UIApplicationDelegate

The App Delegate



AppDelegate.h

```
@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

AppDelegate.m

```
@implementation AppDelegate

– (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {…}

– (void)applicationWillResignActive:(UIApplication *)application {…}

– (void)applicationDidEnterBackground:(UIApplication *)application {…}

– (void)applicationWillEnterForeground:(UIApplication *)application {…}

– (void)applicationDidBecomeActive:(UIApplication *)application {…}

- (void)applicationWillTerminate:(UIApplication *)application {…}

@end
```

AppDelegate.m

```
@implementation AppDelegate

– (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {…}

– (void)applicationWillResignActive:(UIApplication *)application {…}

– (void)applicationDidEnterBackground:(UIApplication *)application {…}

– (void)applicationWillEnterForeground:(UIApplication *)application {…}

– (void)applicationDidBecomeActive:(UIApplication *)application {…}

- (void)applicationWillTerminate:(UIApplication *)application {…}

@end
```

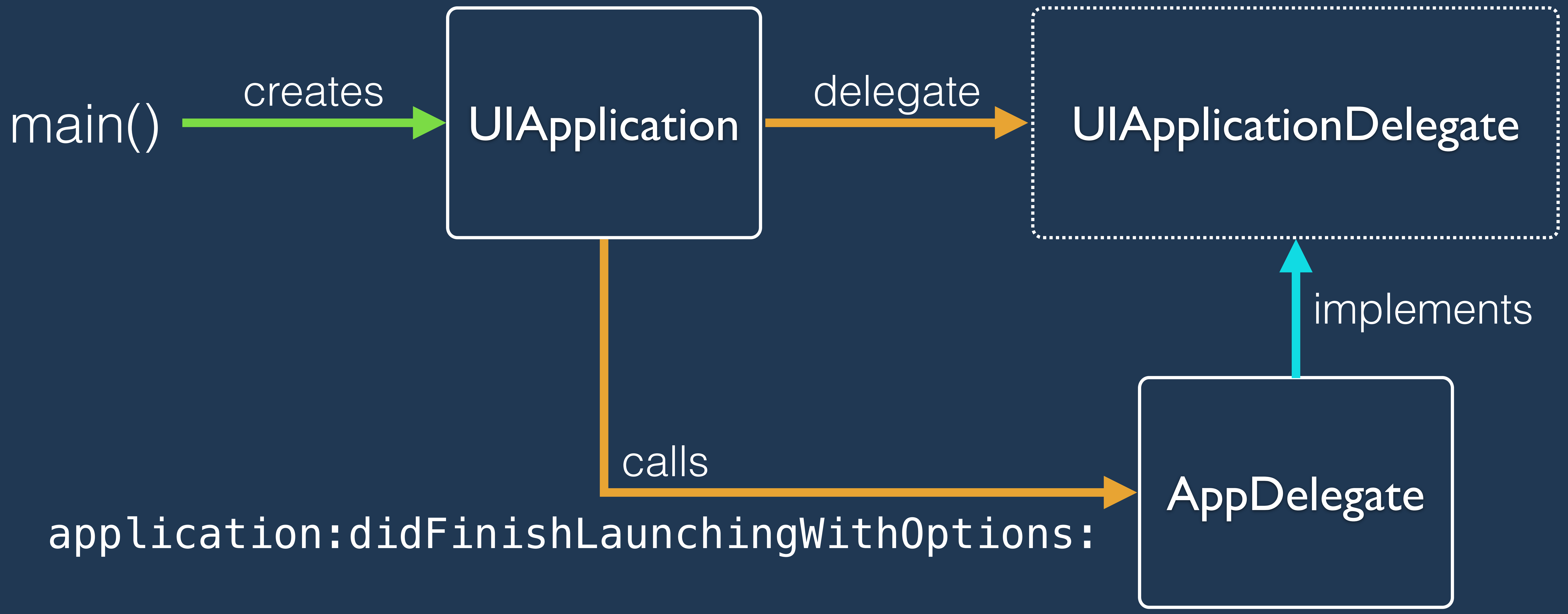
AppDelegate.m

For now, this is the only method we care about

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {…}
```

This is the first method called after startup and the place where we put our first lines of custom code.

The App Delegate



Window what?

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

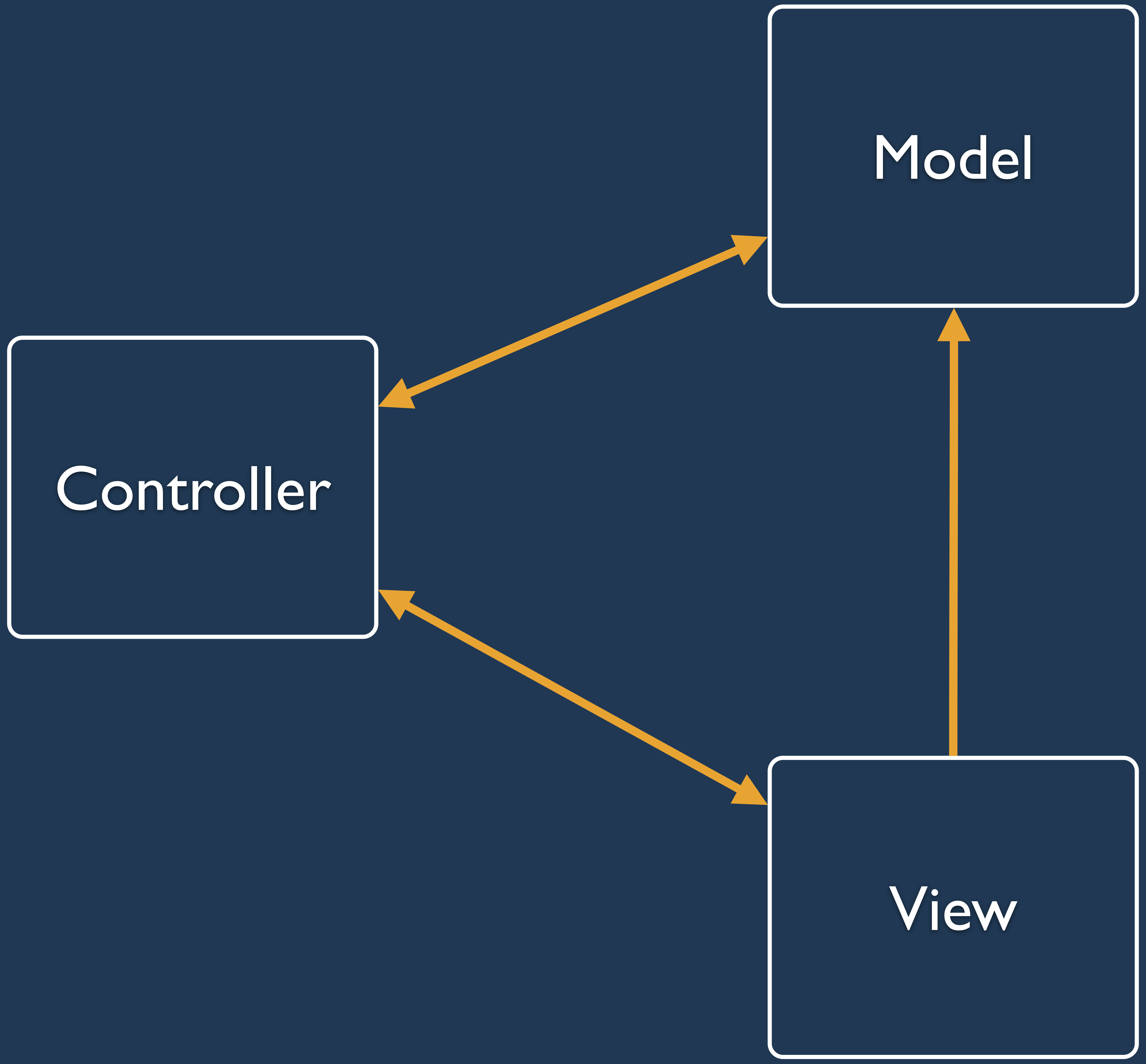
Root who?

```
Smarticle[44114:60b] Application windows are expected to
have a root view controller at the end of application
launch
```

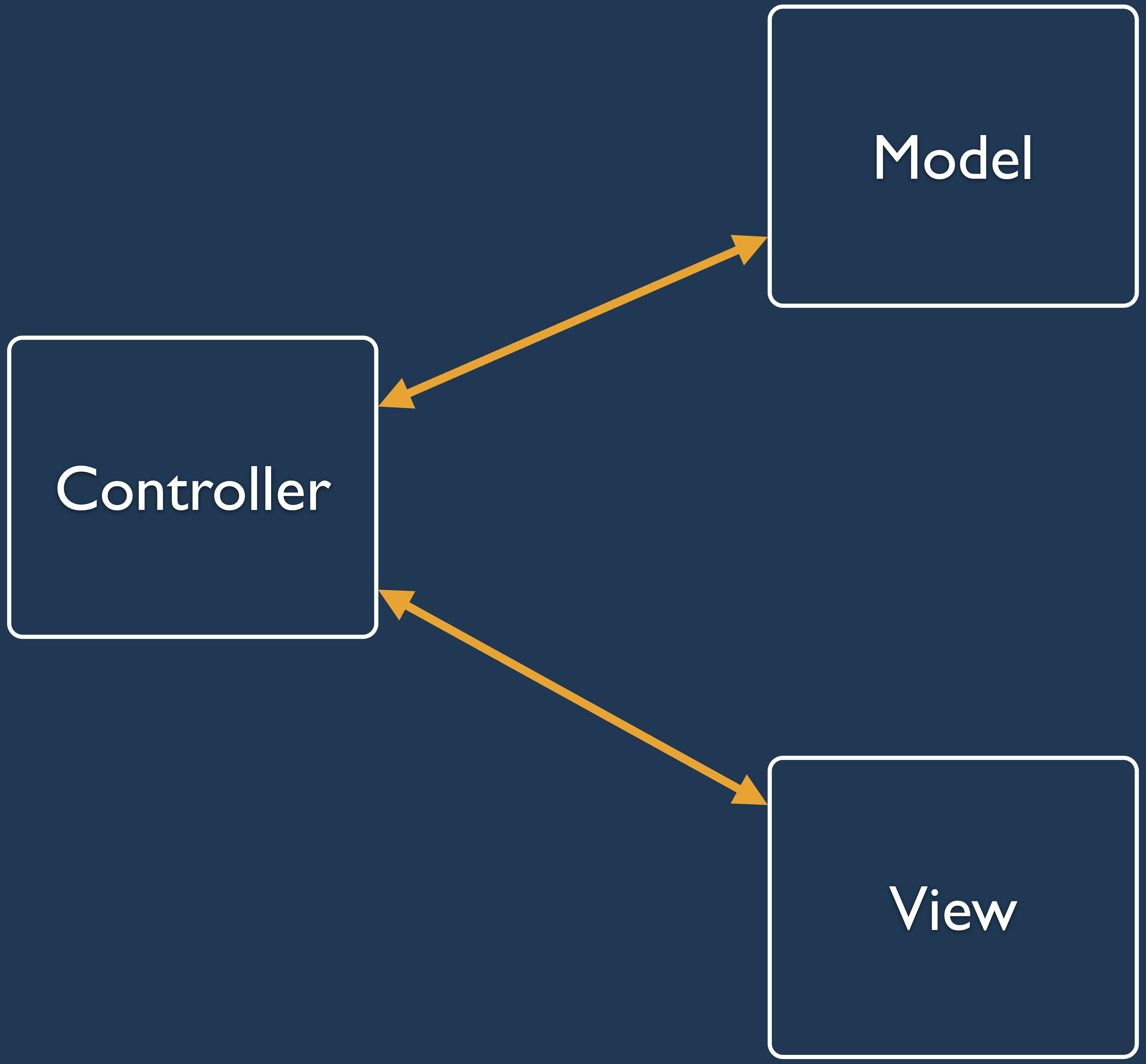
UIWindow
UIViewController

Model View Controller

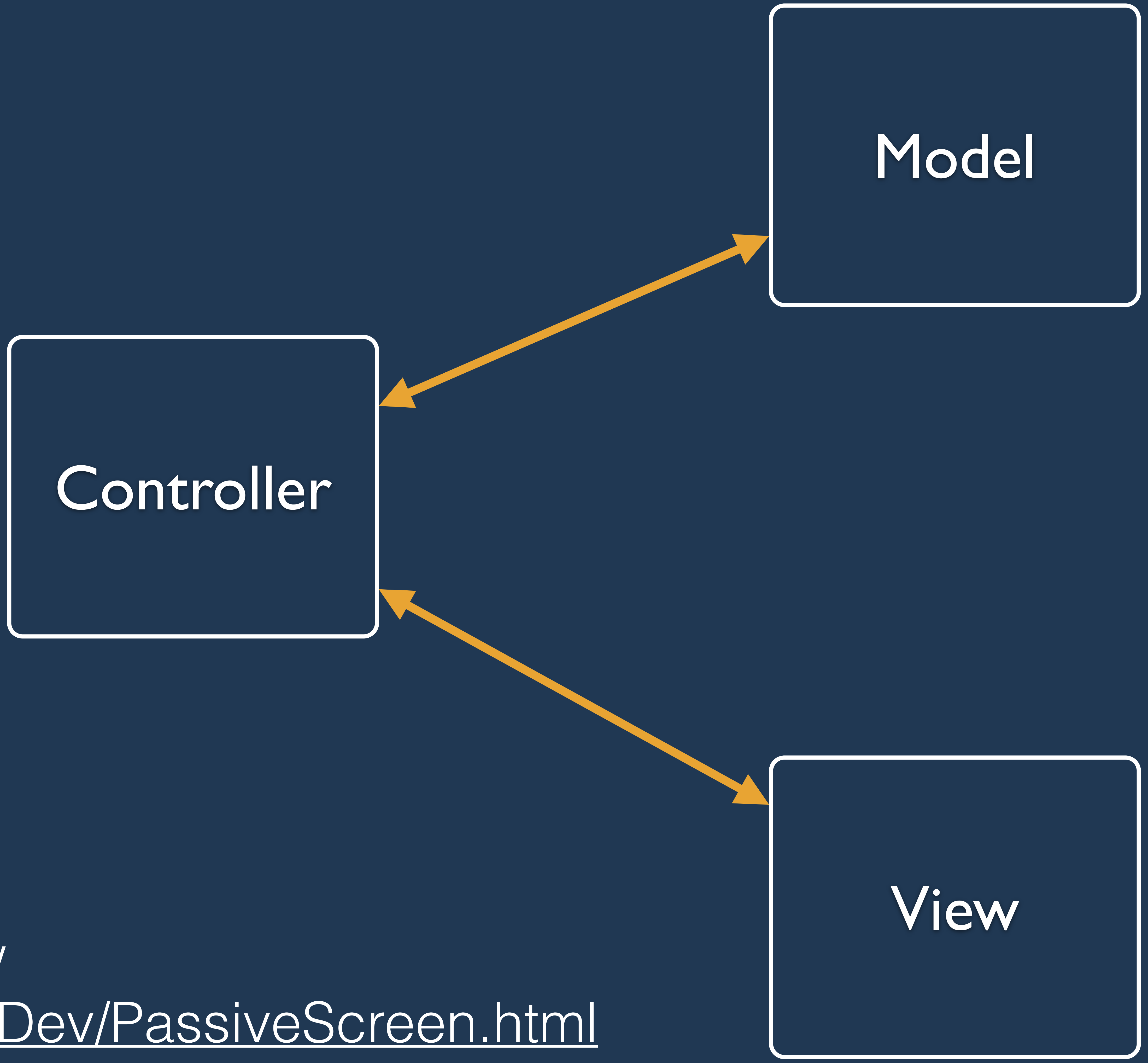
Classic MVC



iOS MVC

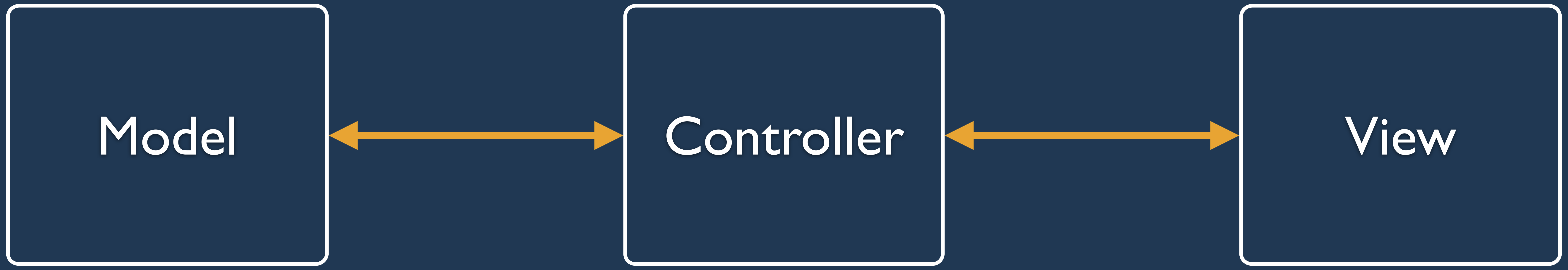


iOS MVC

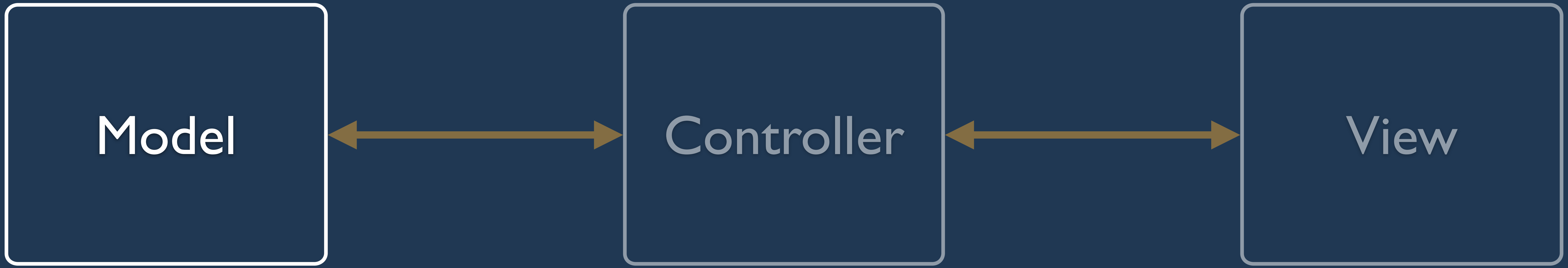


Martin Fowler, Passive View
http://martinfowler.com/eaaDev/PassiveScreen.html

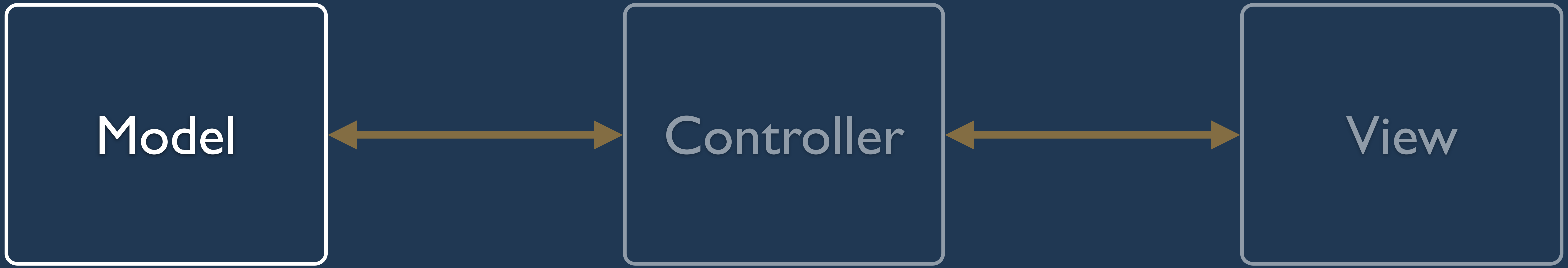
iOS MVC



iOS MVC

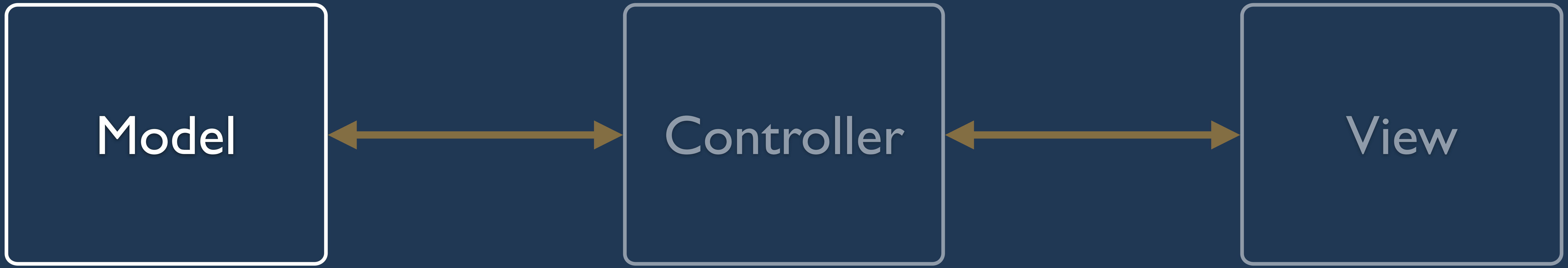


iOS MVC



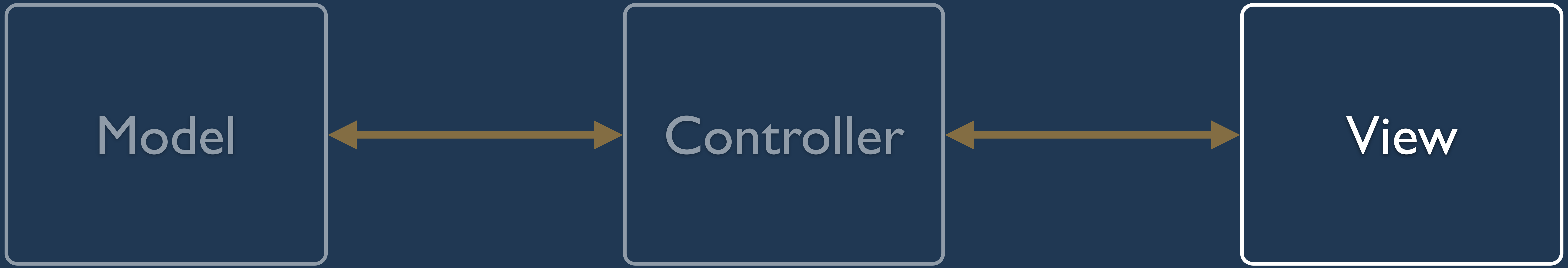
- Pure representation of data and relationships between data

iOS MVC



- Pure representation of data and relationships between data
- Methods for retrieving that data

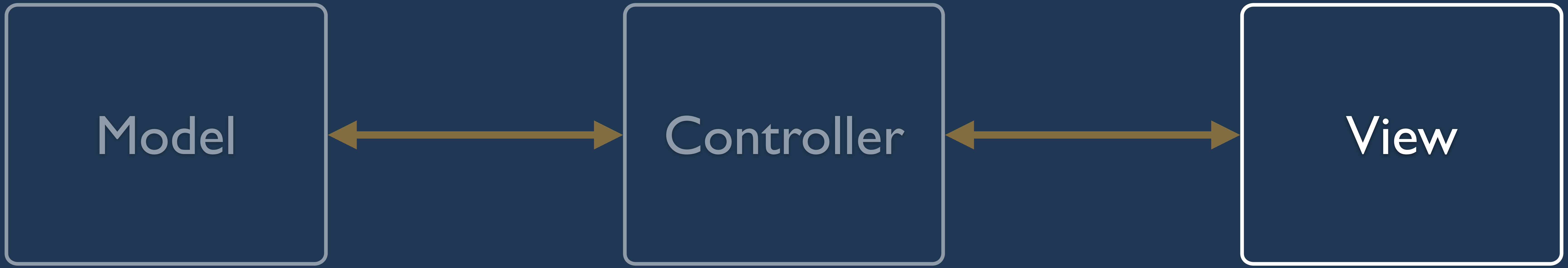
iOS MVC


Model

Controller

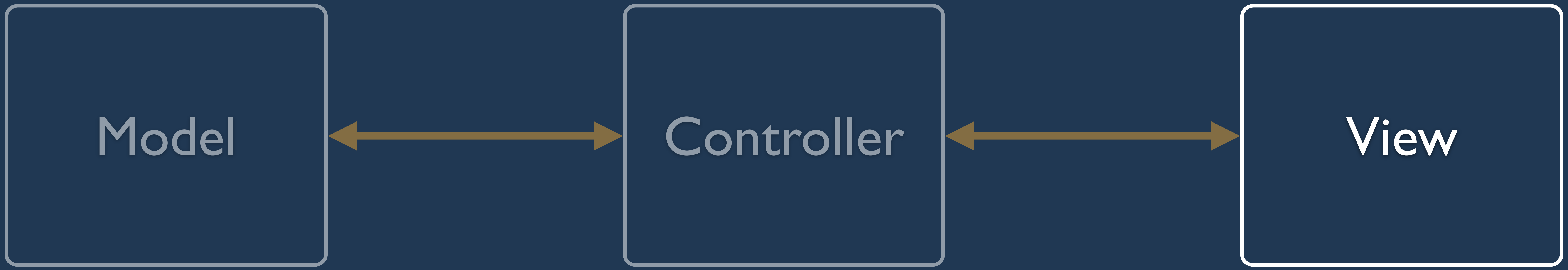
View

iOS MVC



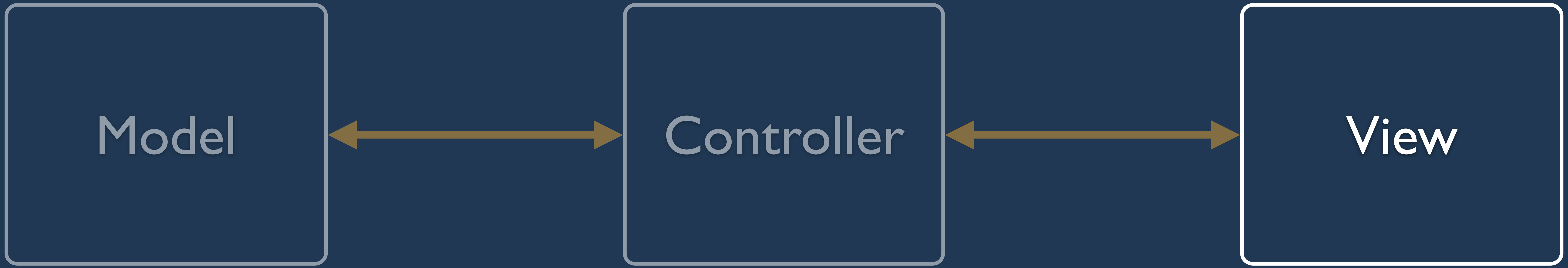
- Represents a visible area of the screen

iOS MVC



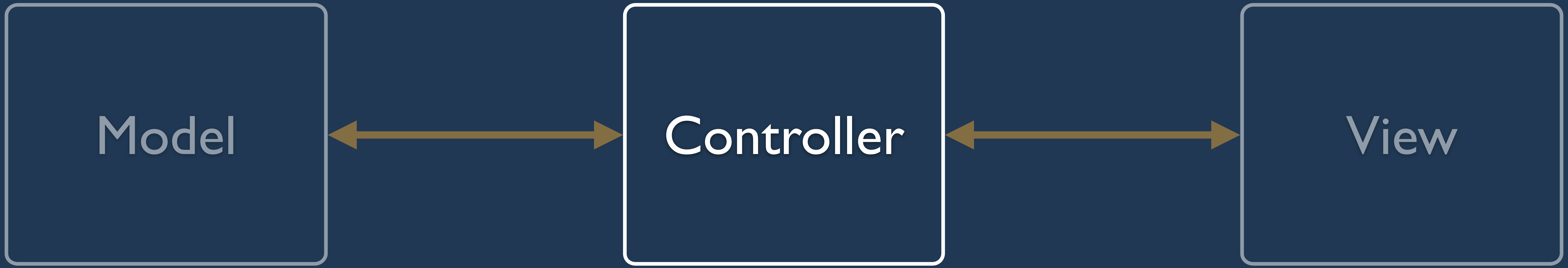
- Represents a visible area of the screen
- Interprets events for that area of the screen

iOS MVC

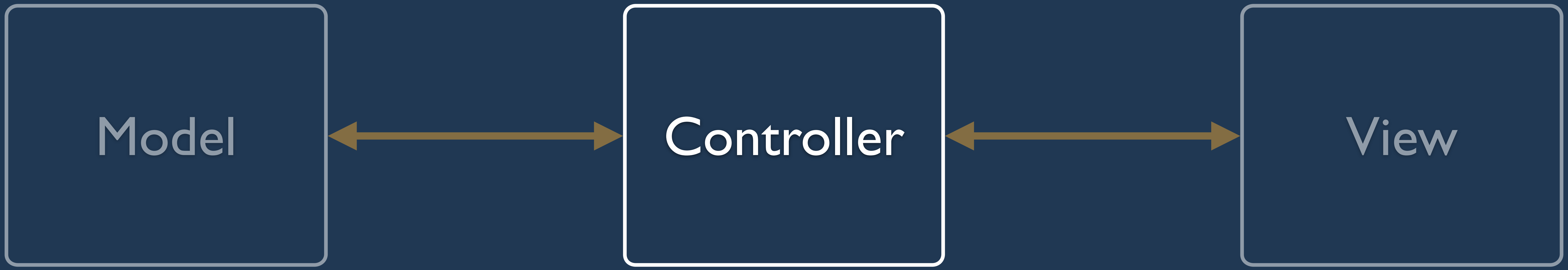


- Represents a visible area of the screen
- Interprets events for that area of the screen
- Views can contain other views

iOS MVC

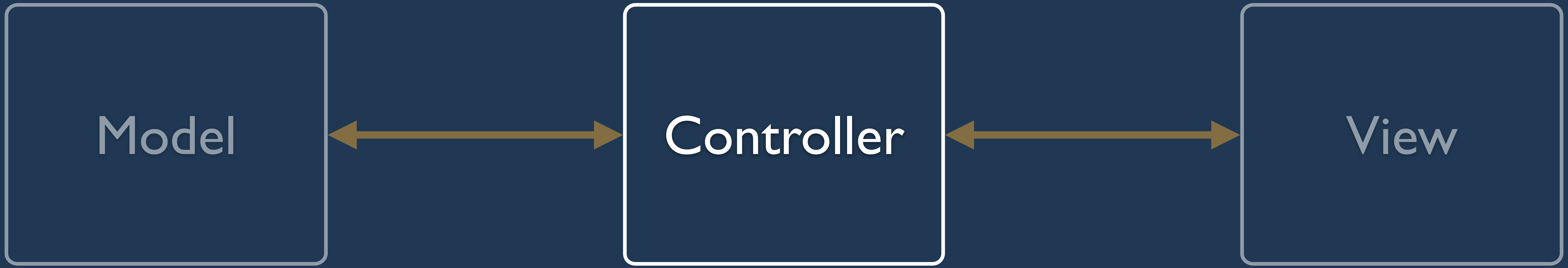


iOS MVC



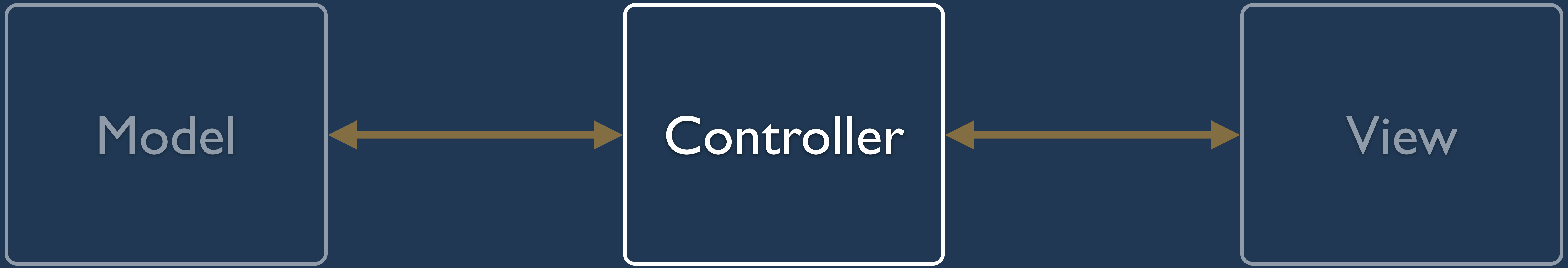
- Updates and observes the model

iOS MVC



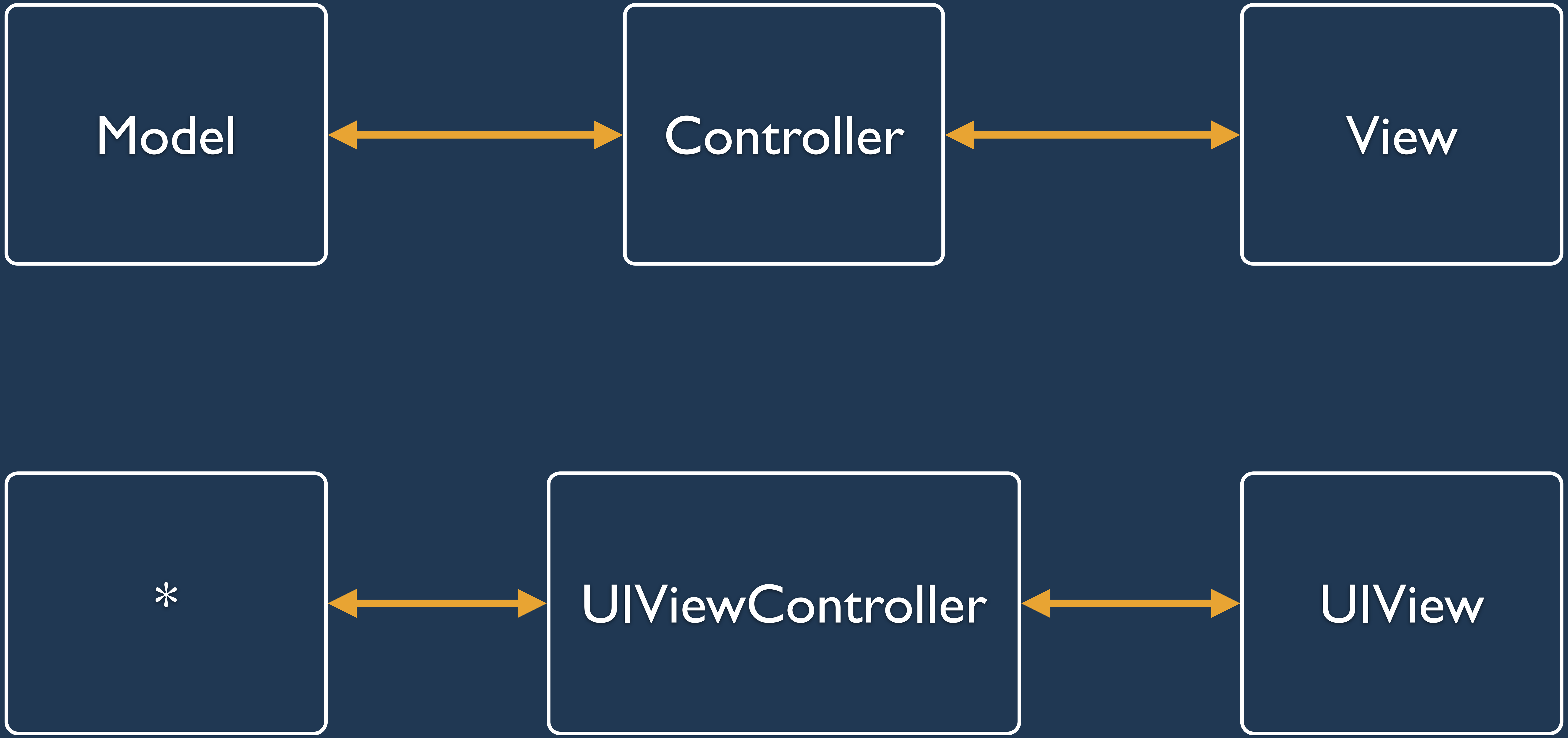
- Updates and observes the model
- Updates and observes the view

iOS MVC



- Updates and observes the model
- Updates and observes the view
- The app’s behavior - brains

iOS MVC



Model

Controller

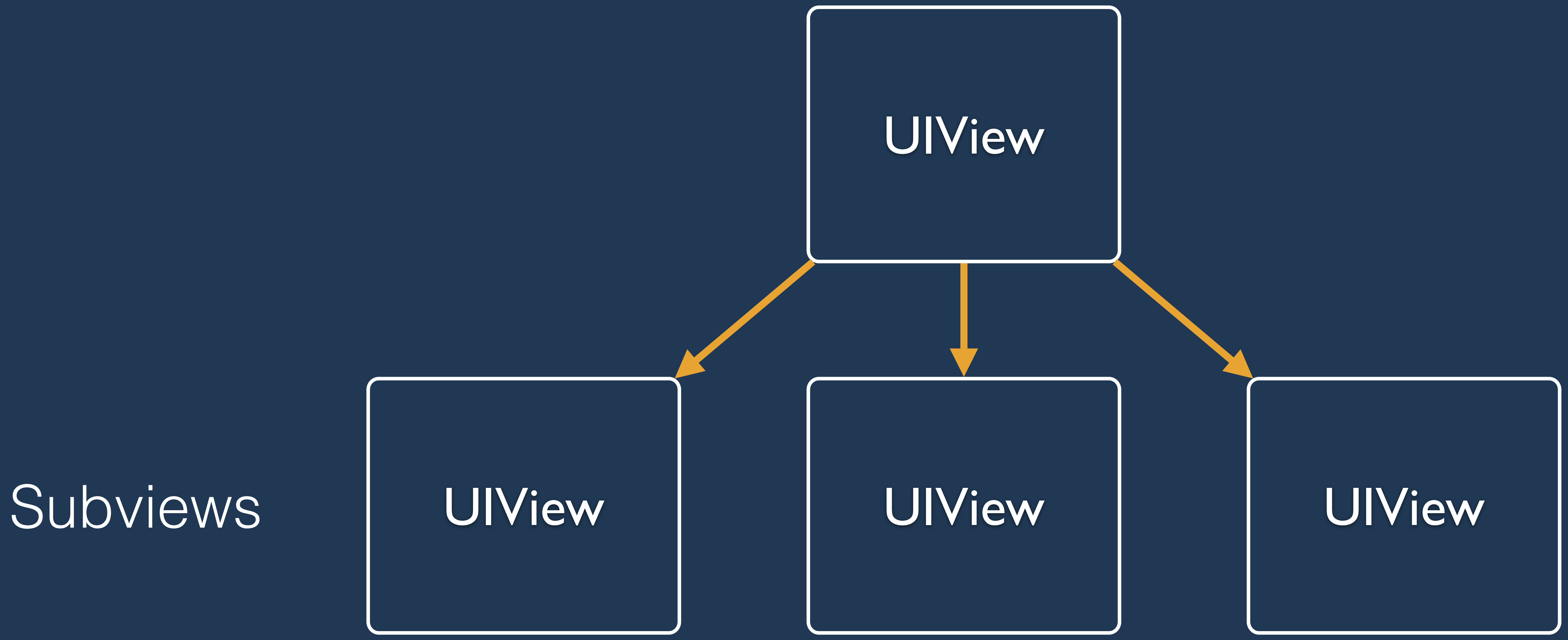
View

*

UIViewController

UIView

UIView



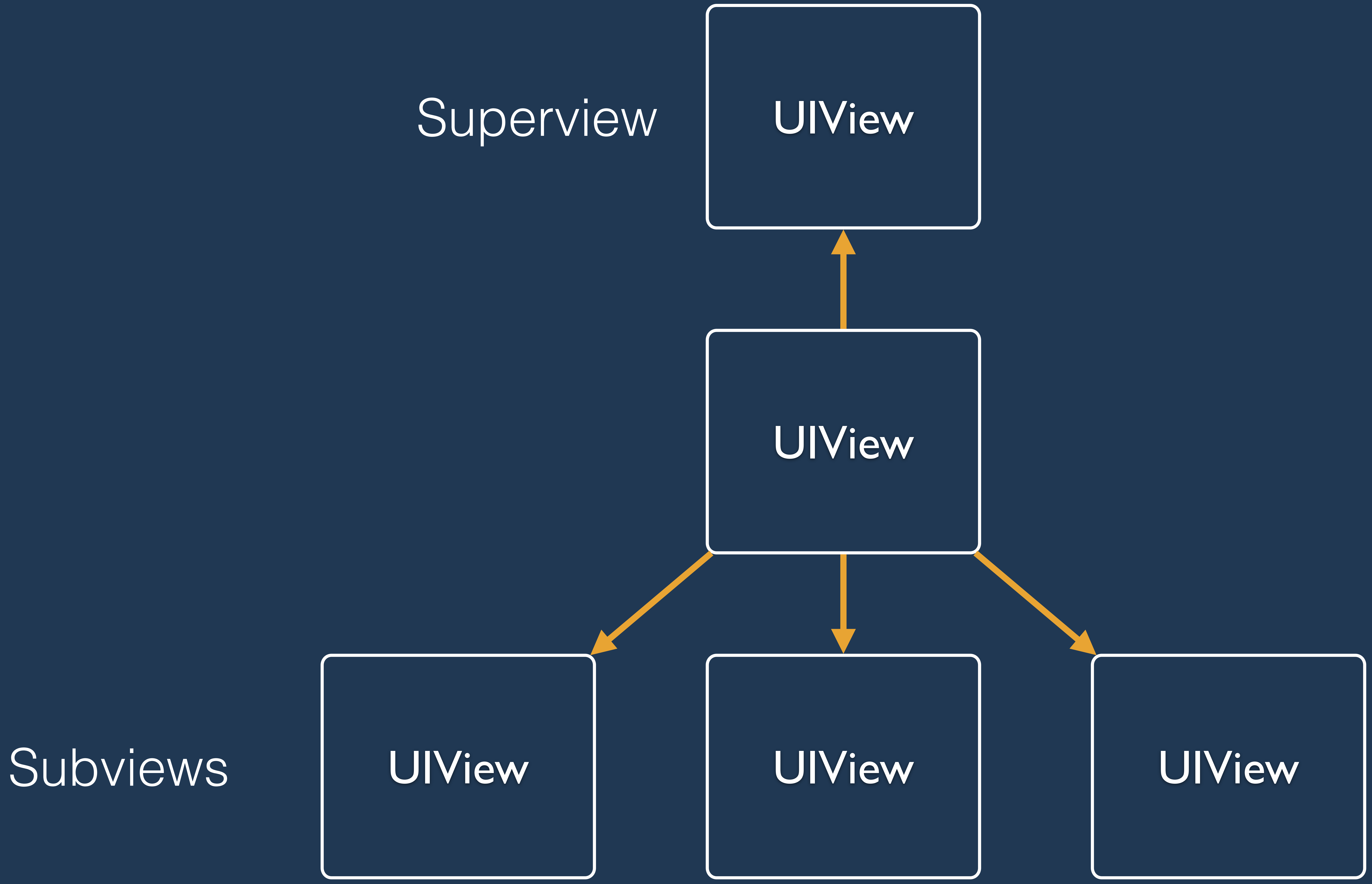
UIView

Subviews

UIView

UIView

UIView



Superview

UIView

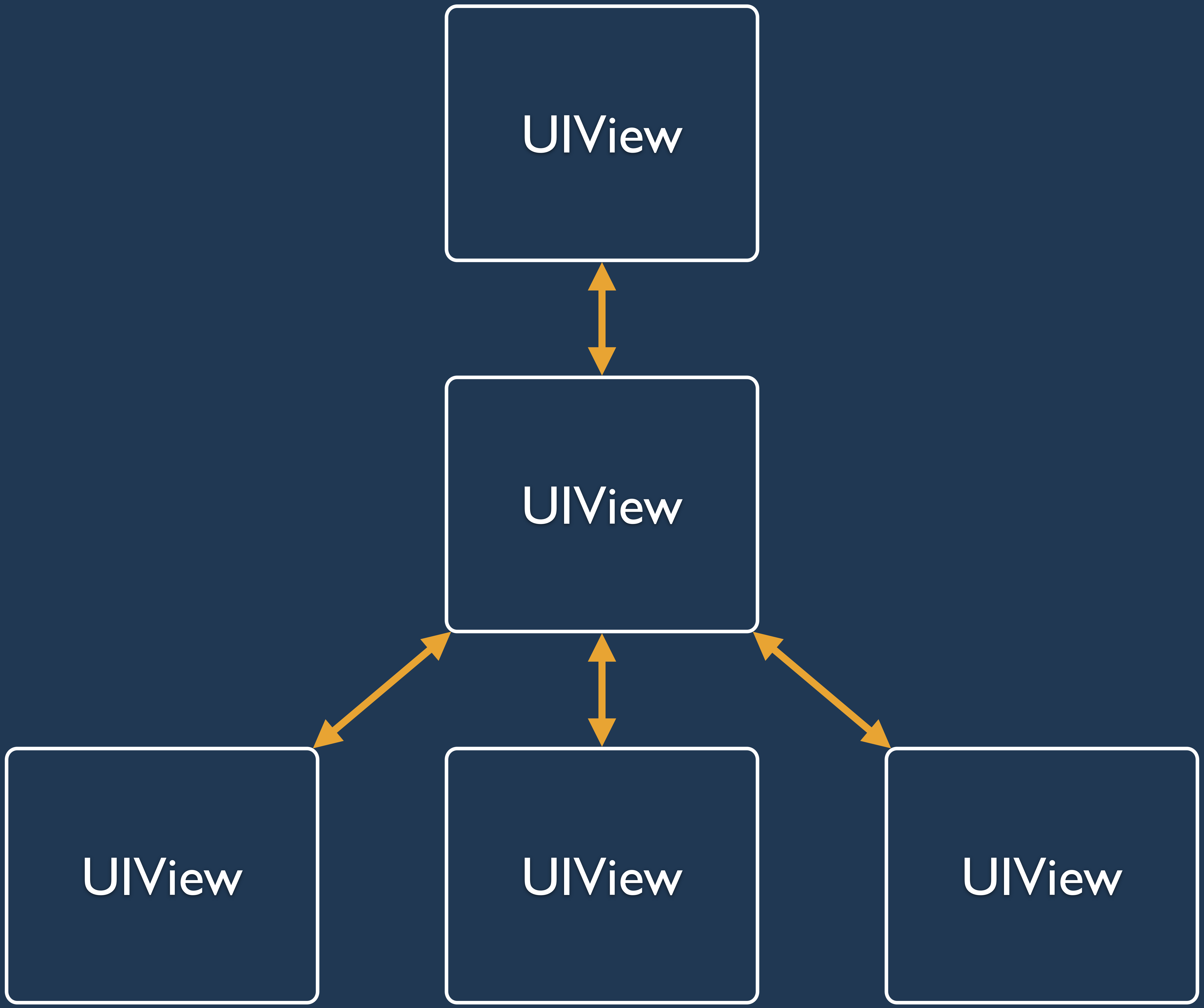
UIView

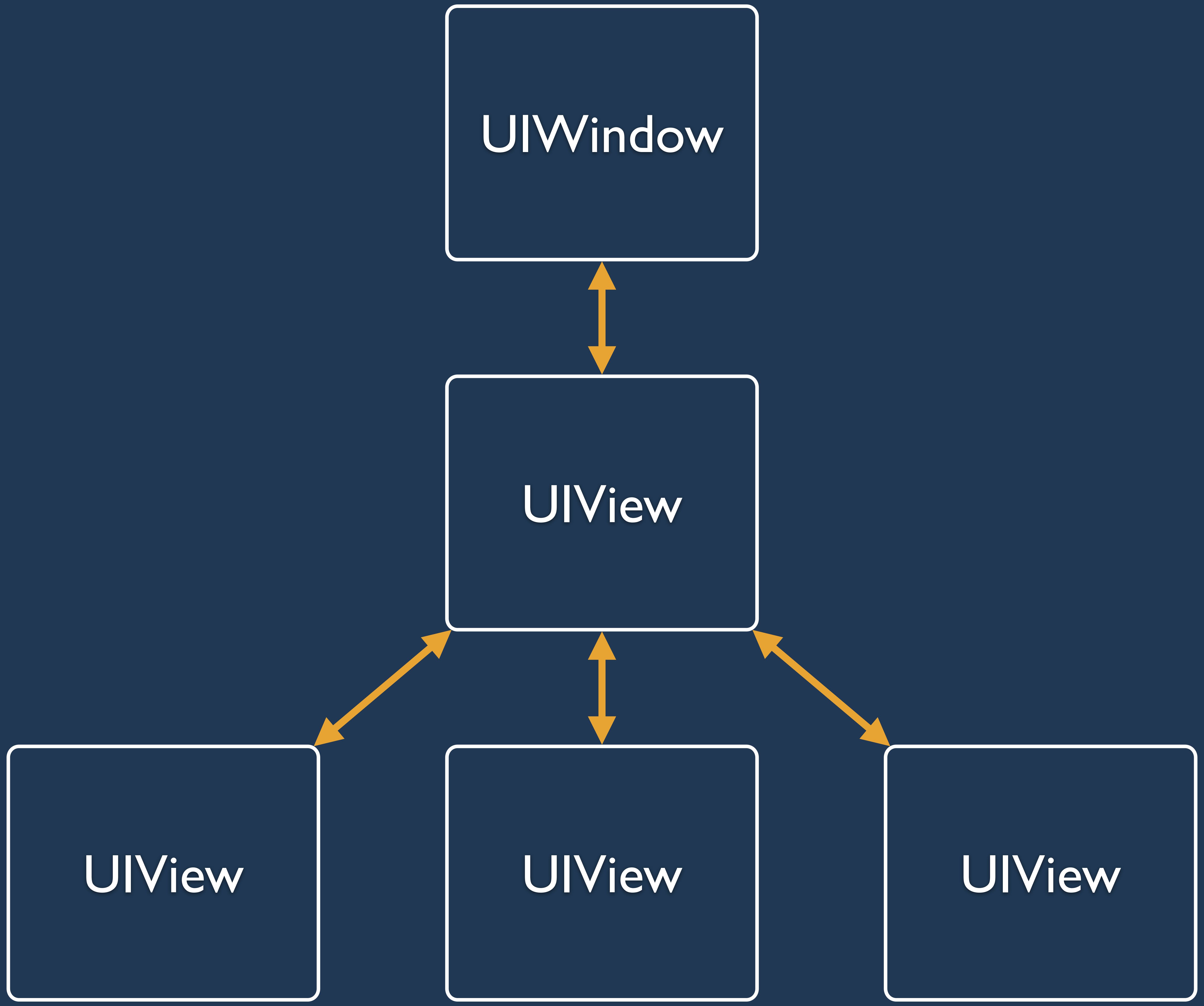
Subviews

UIView

UIView

UIView


UIView
UIView
UIView
UIView
UIView

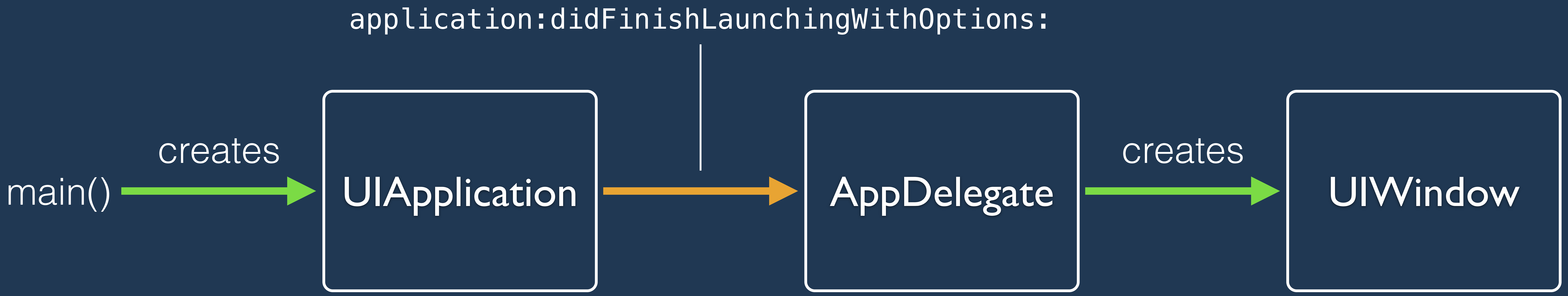

UIWindow
UIView
UIView
UIView
UIView

UIWindow

- A special type of UIView that contains all other views
- Always at the root of the view hierarchy
- An iOS app only has one window (unless it supports external screens.)

Now we know what some of this means

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```



application:didFinishLaunchingWithOptions:

main()

creates

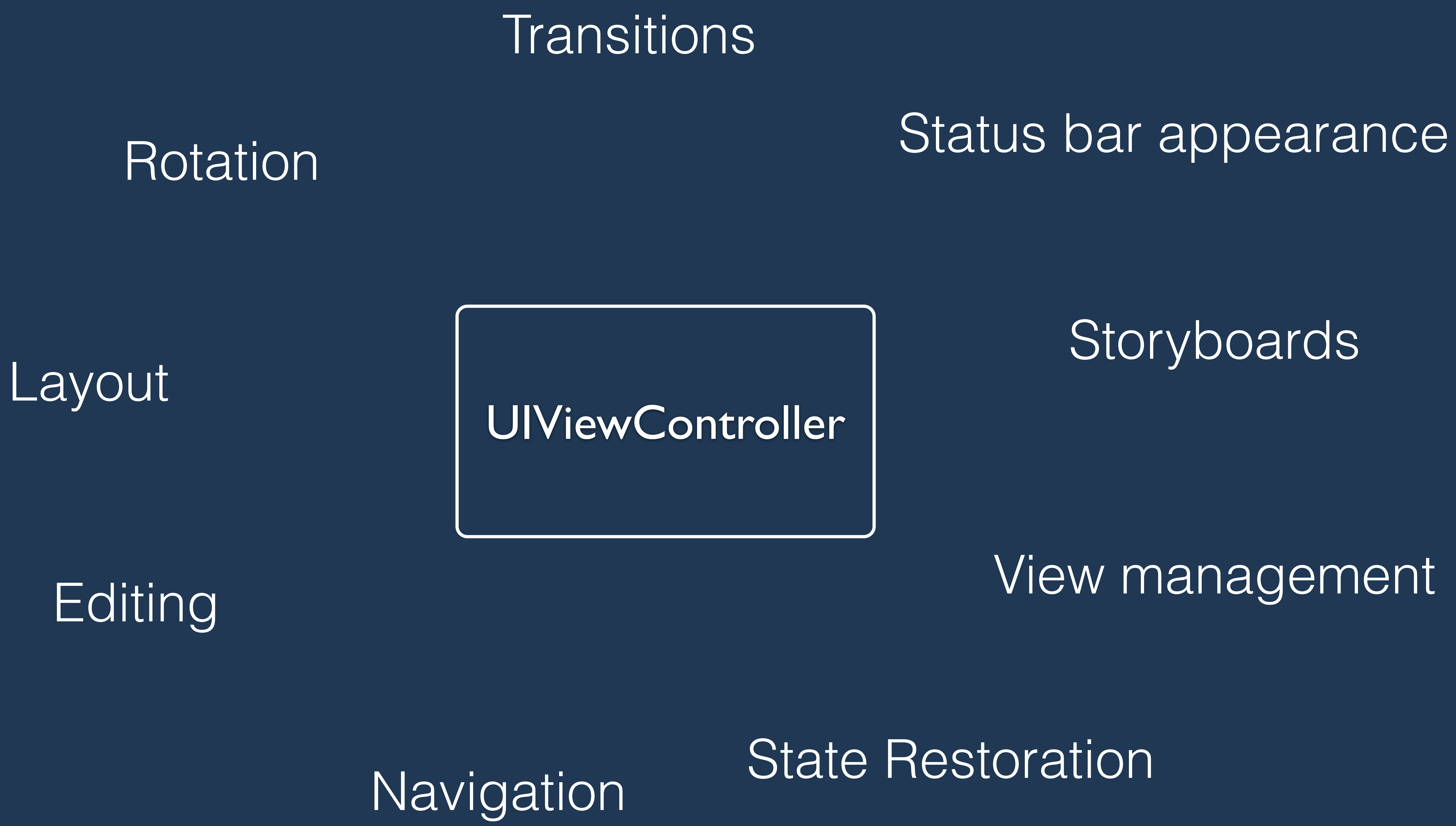
UIApplication

AppDelegate

creates

UIWindow

UIViewController



Transitions

Rotation

Status bar appearance

Layout

UIViewController

Storyboards

Editing

View management

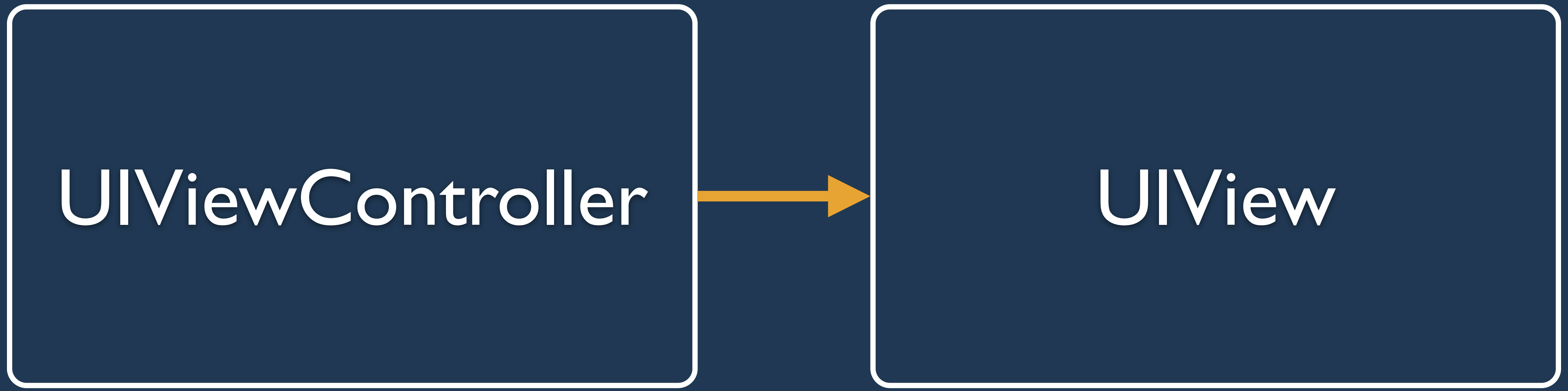
State Restoration

Navigation

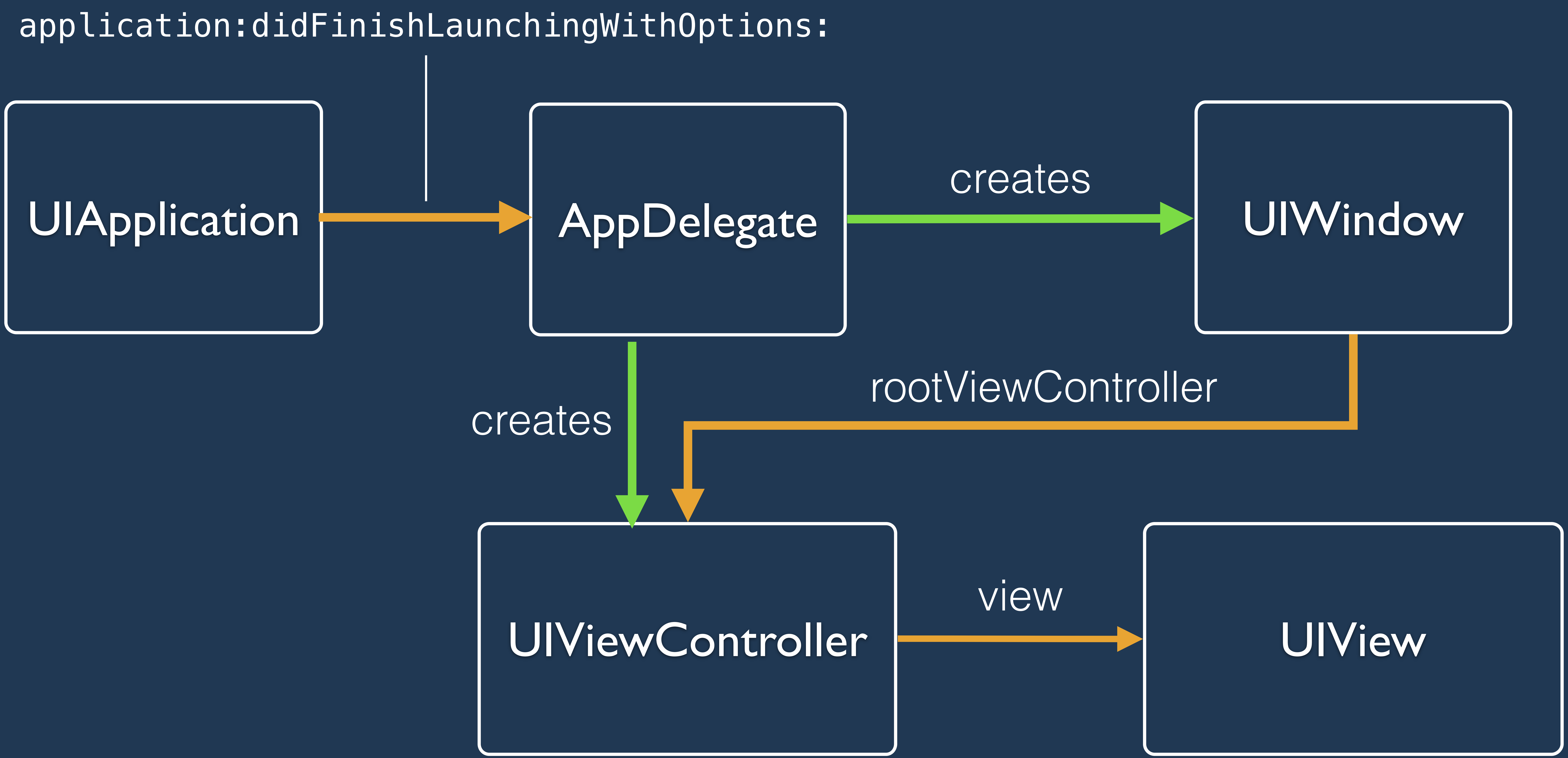
View management

UIViewController

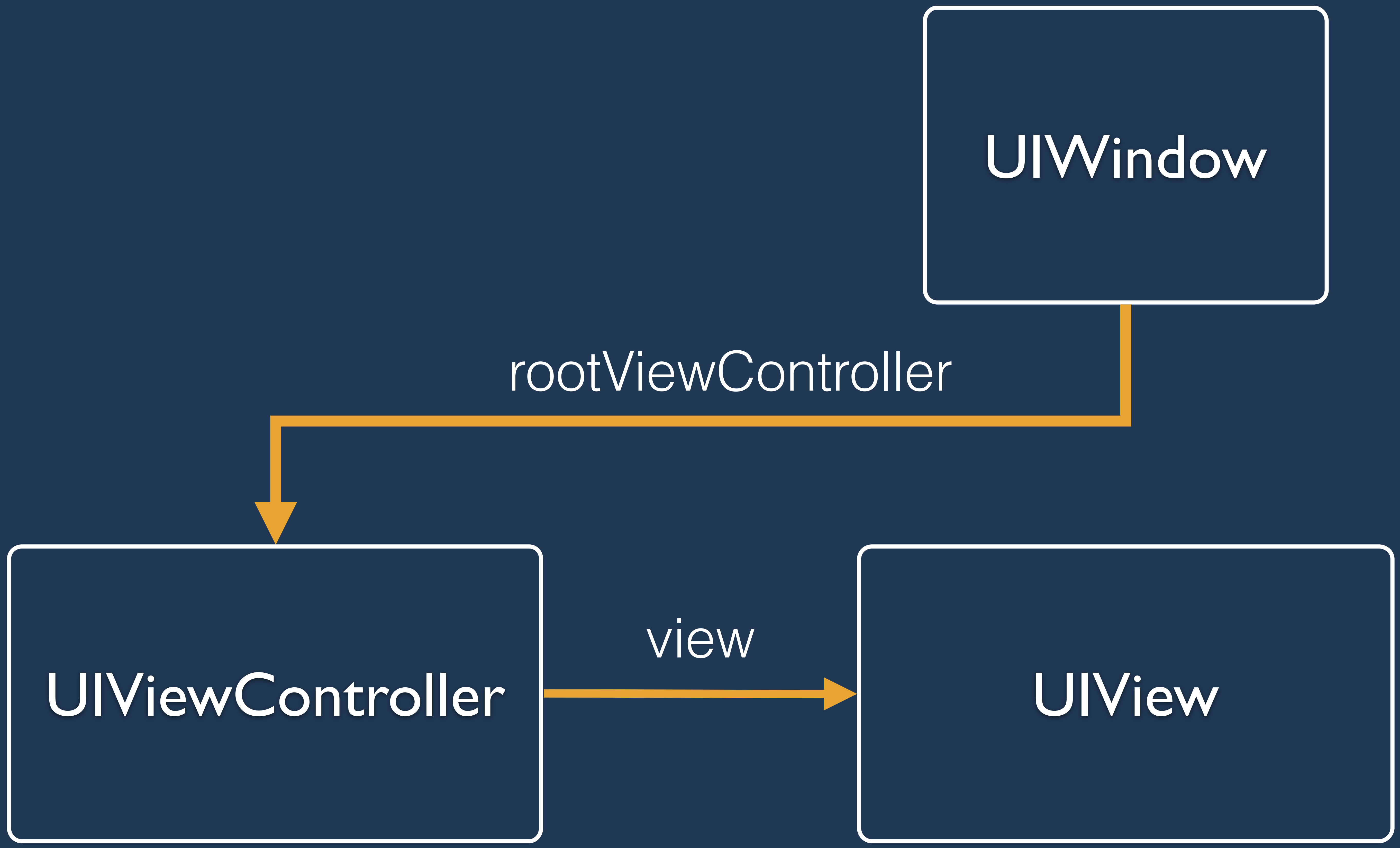
View management



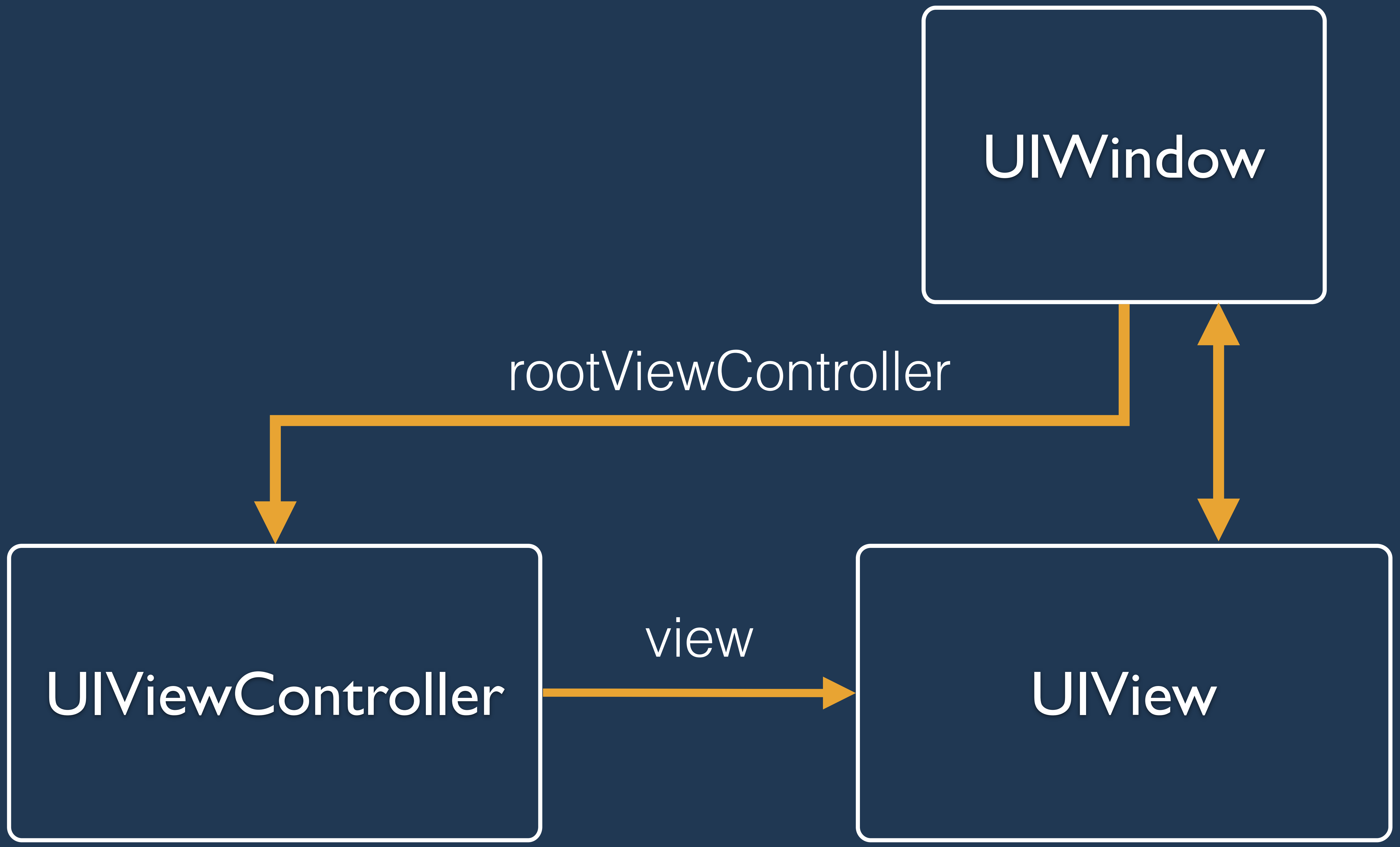
Each view controller manages one content view (and its subviews)


application:didFinishLaunchingWithOptions:
UIApplication
AppDelegate
creates
UIWindow
rootViewController
creates
UIViewController
view
UIView

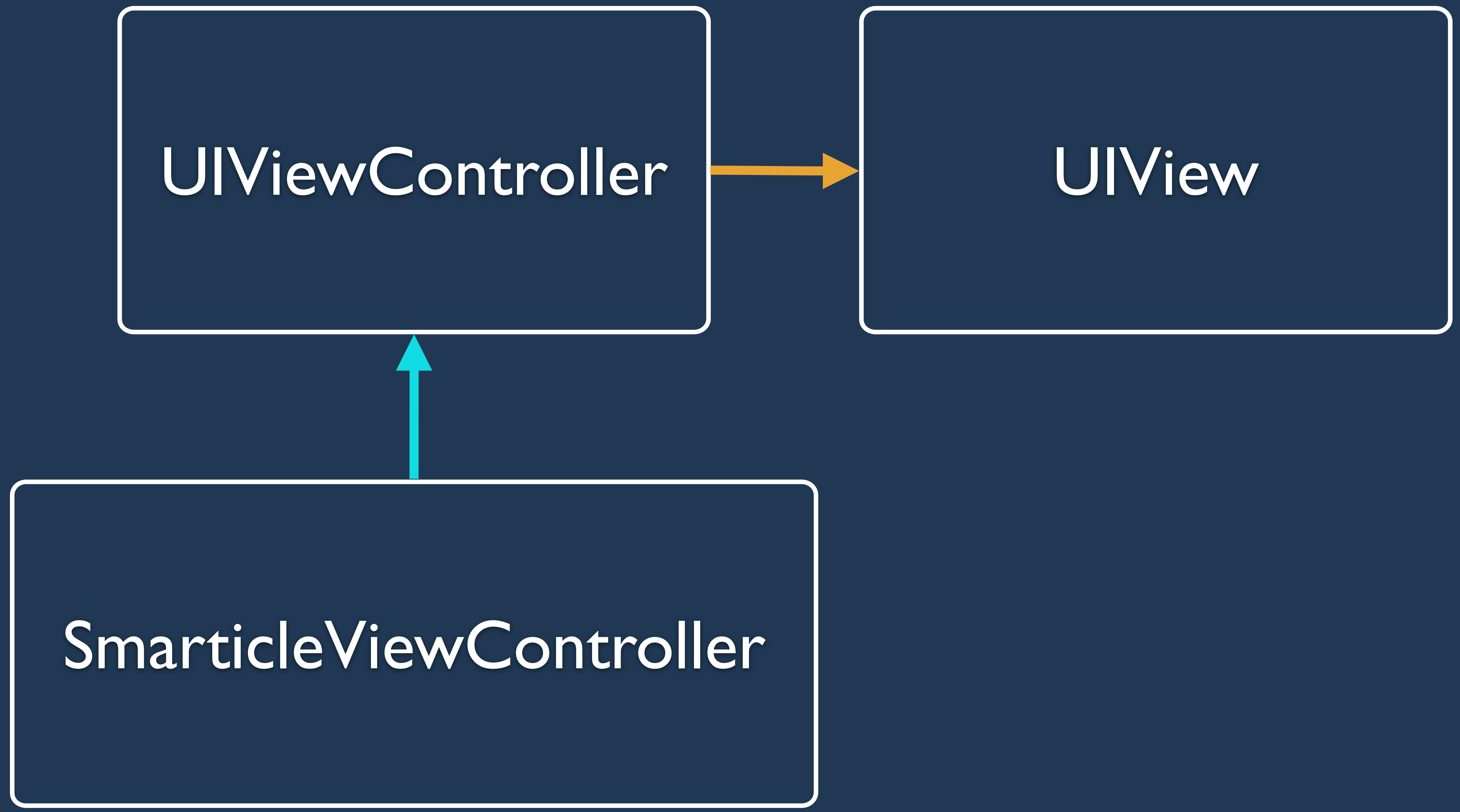
Setting the rootViewController on the window magically adds its view as a subview



Setting the rootViewController on the window magically adds its view as a subview

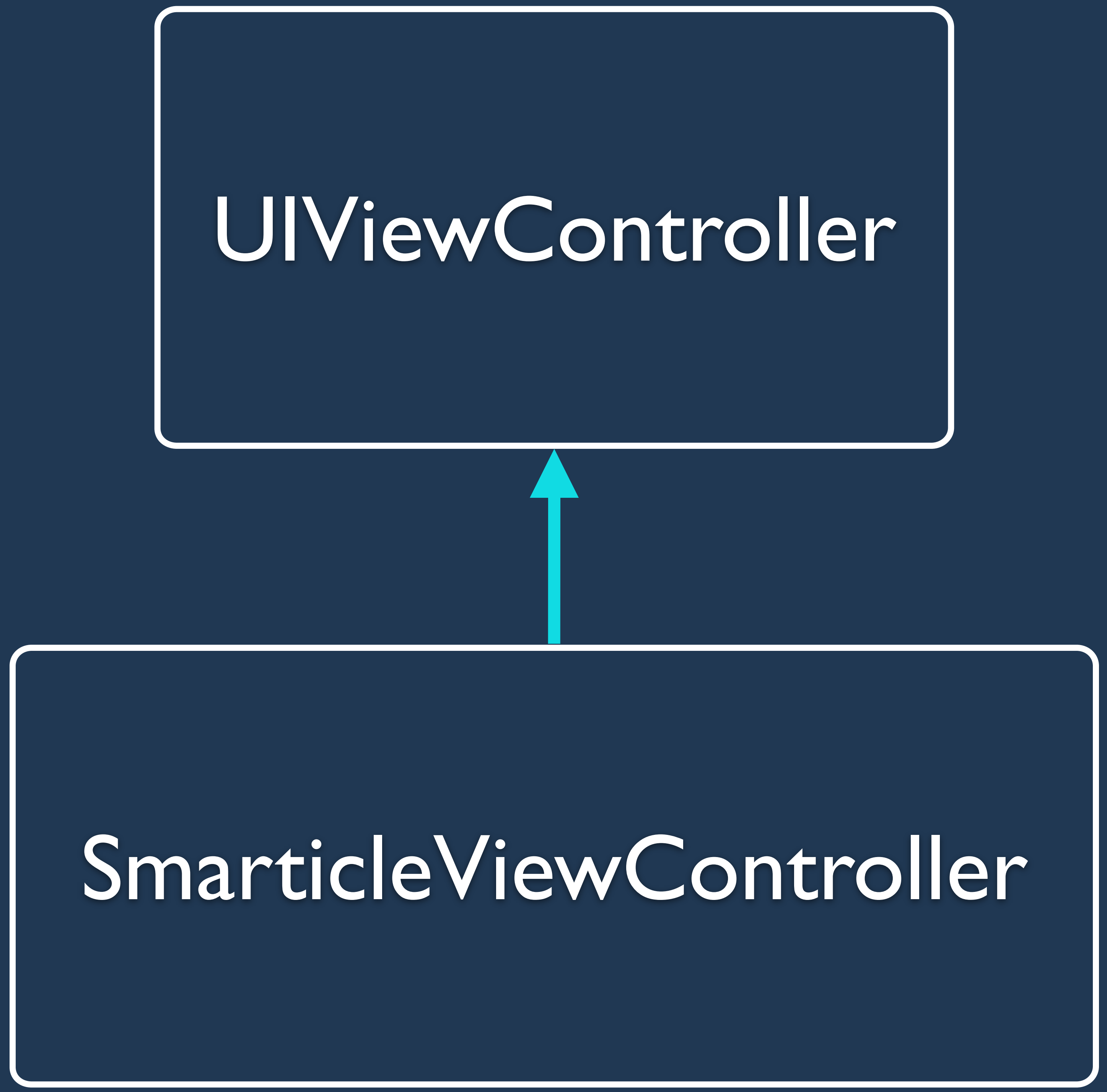


By default, a regular view controller displays an empty view



To customize, create a subclass

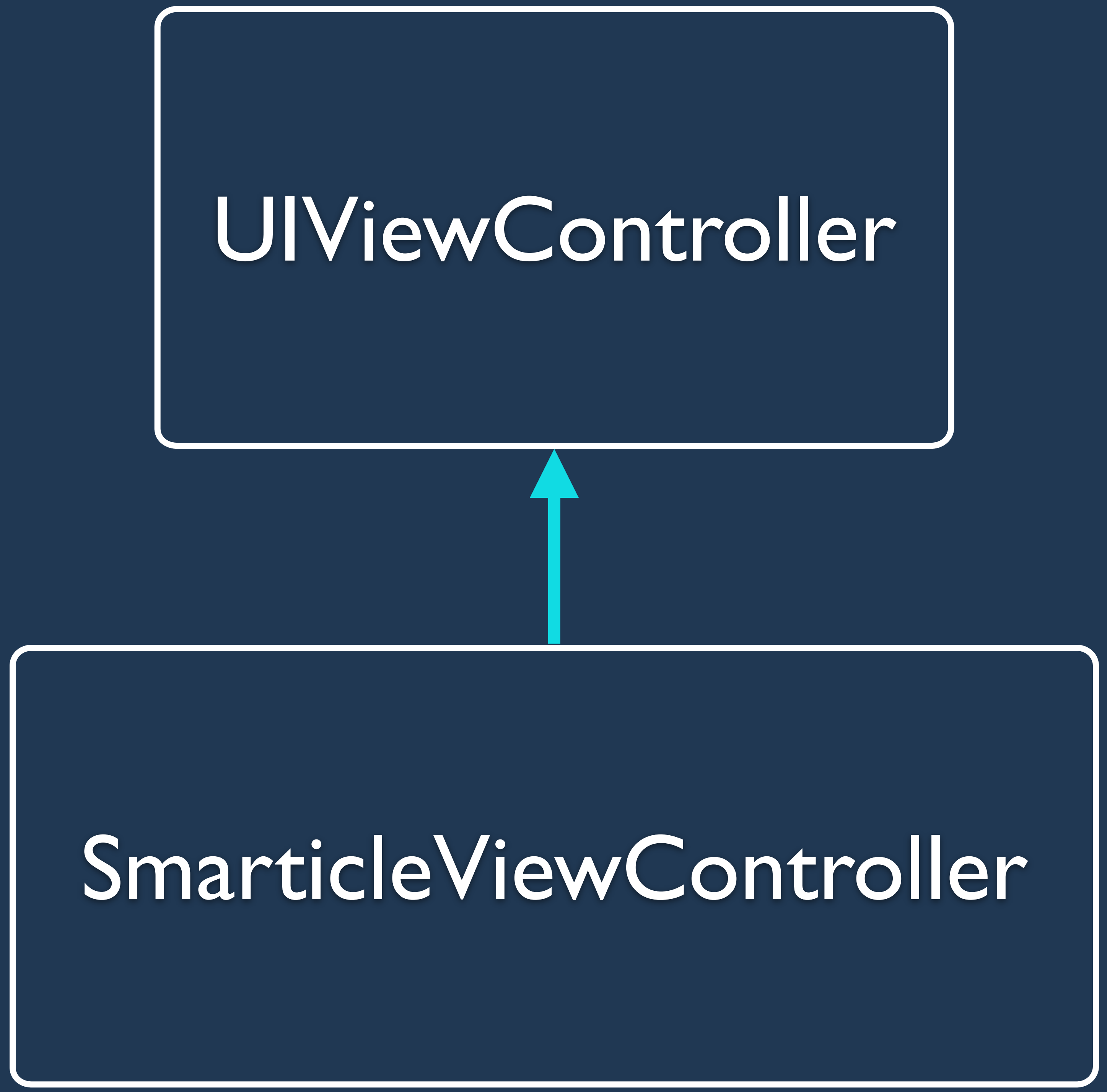
UIViewController subclass



```
// Default value is nil
@property(nonatomic,retain) UIView *view;
```

A view controller’s view is loaded on demand, the first time the view property is accessed.

UIViewController subclass



```
// Default value is nil
@property(nonatomic,retain) UIView *view;
```

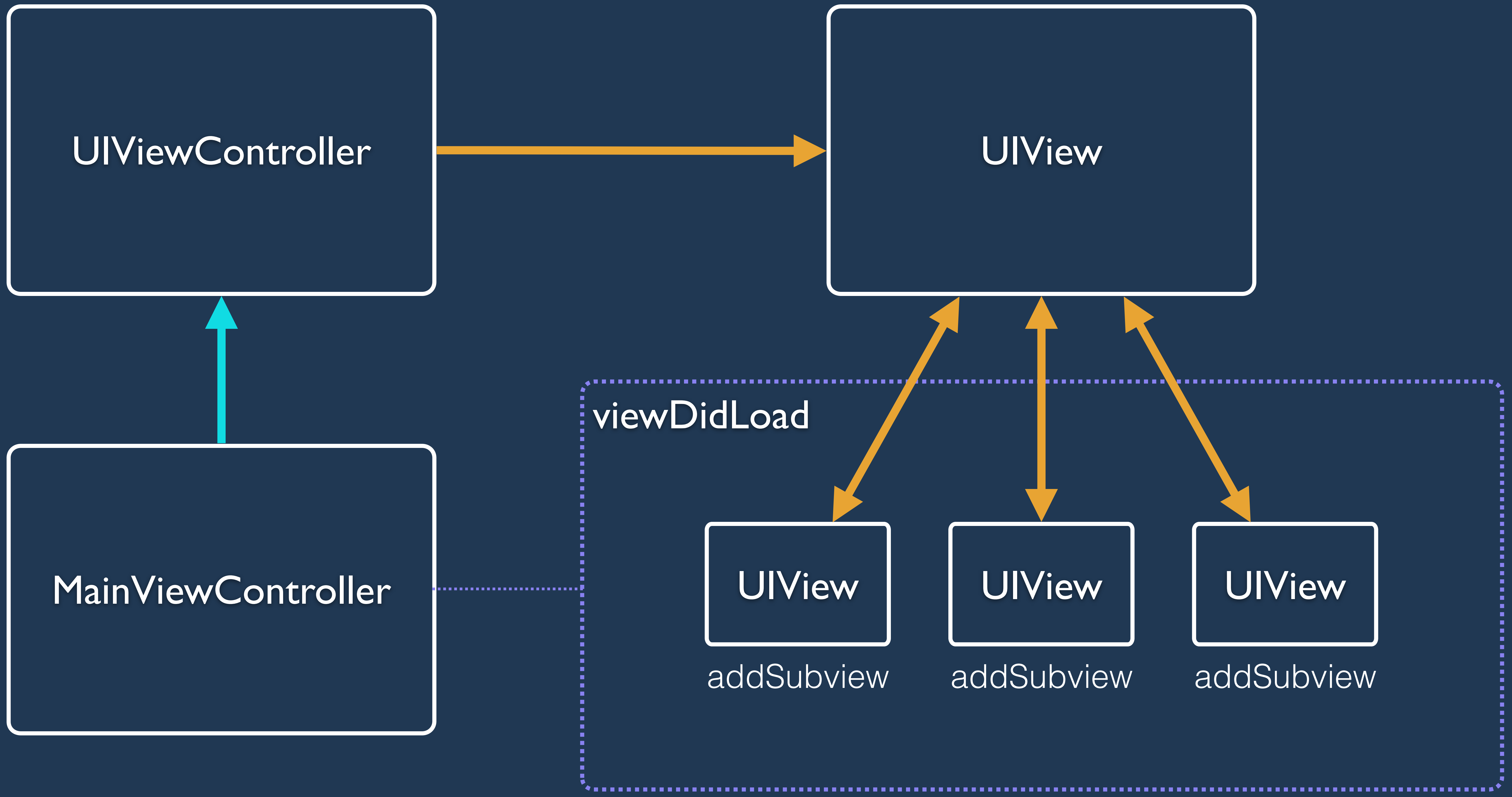
1. Override viewDidLoad, adding subviews
2. Load view hierarchy from a .xib file
3. Load view hierarchy from a storyboard file

viewDidLoad

viewDidLoad

- Called immediately after the default empty UIView is initialized
- Create and configure additional views
- Add those views to self.view

viewDidLoad



viewDidLoad

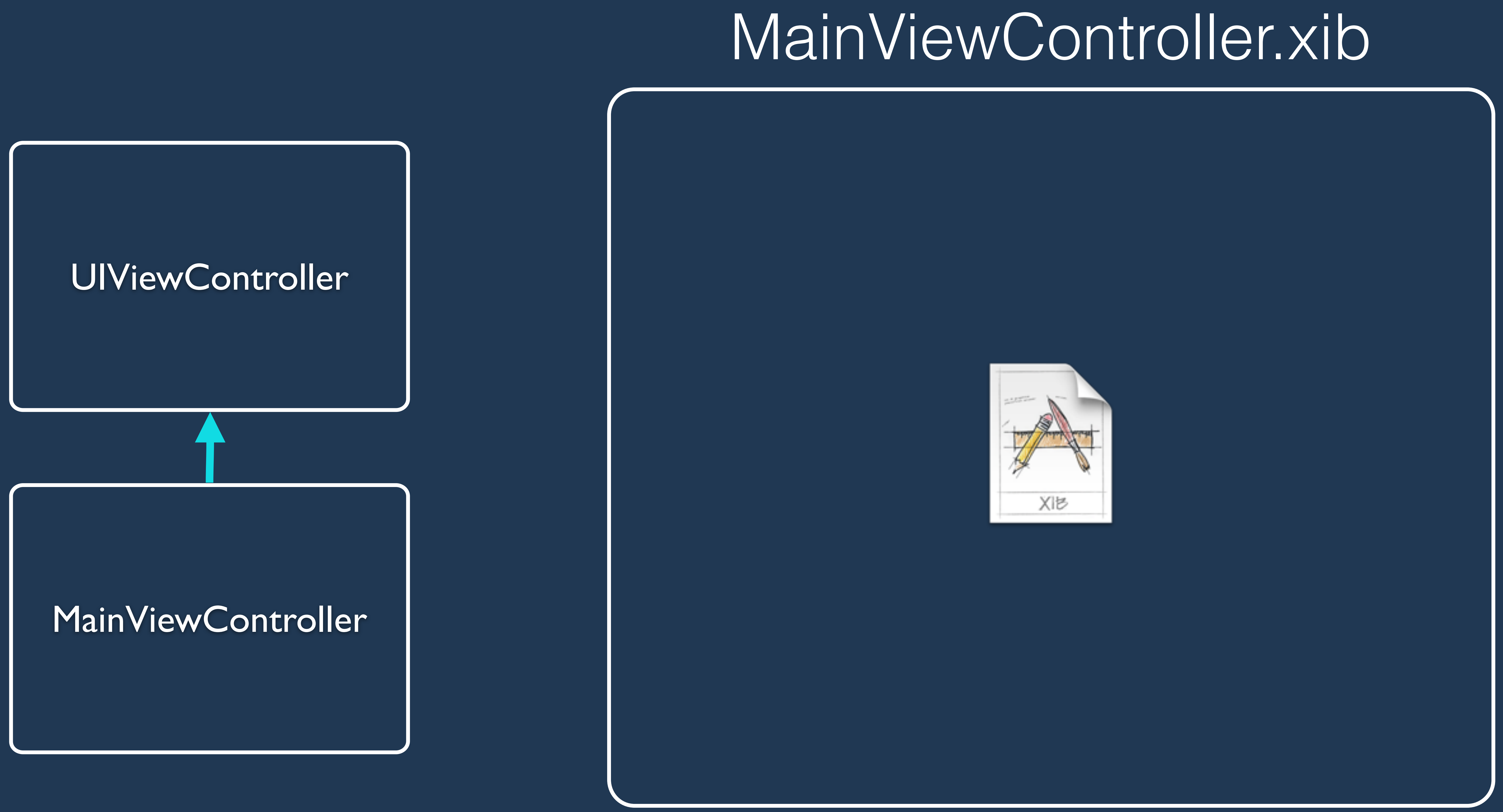
```
MainViewController *main = [[MainViewController alloc] init];
```

.xib file

.xib file

- An opaque XML file that encodes views and their properties
- Allows connections to/from the instantiating view controller via the File Owner placeholder
- At runtime, views are rehydrated and connections are recreated

.xib file

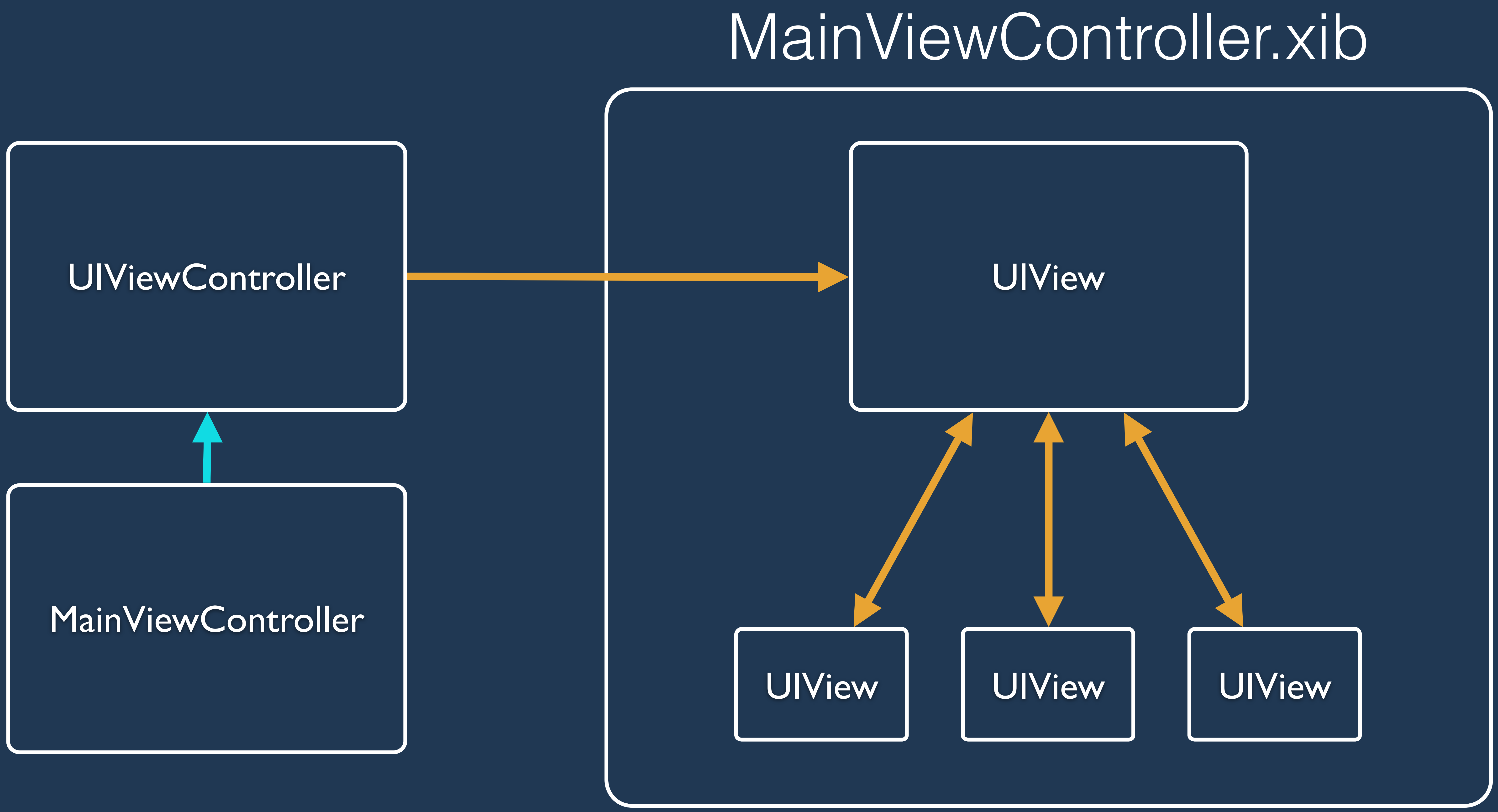


UIViewController

MainViewController

MainViewController.xib

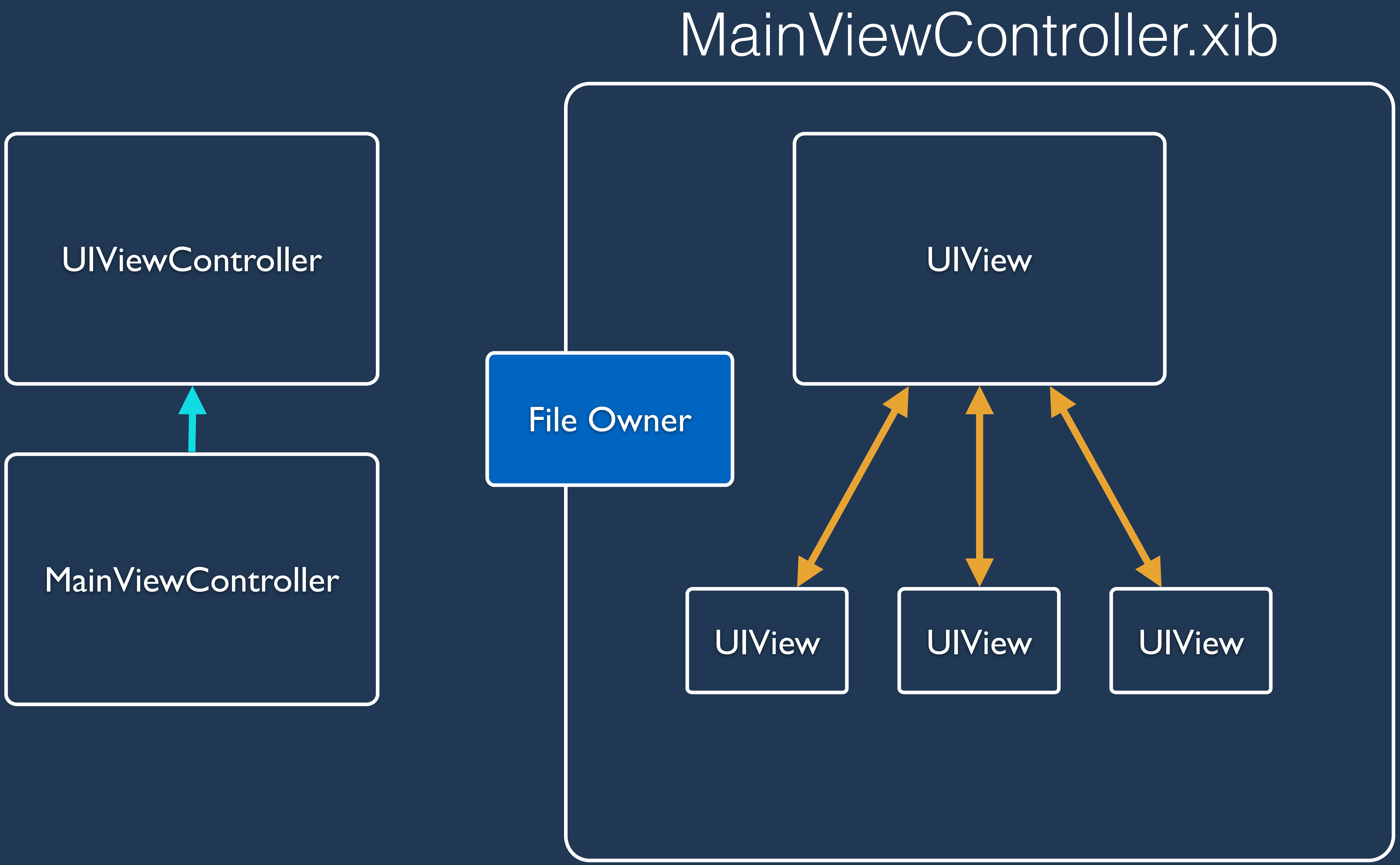
.xib file


MainViewController.xib
UIViewController
MainViewController
UIView
UIView
UIView
UIView

.xib file

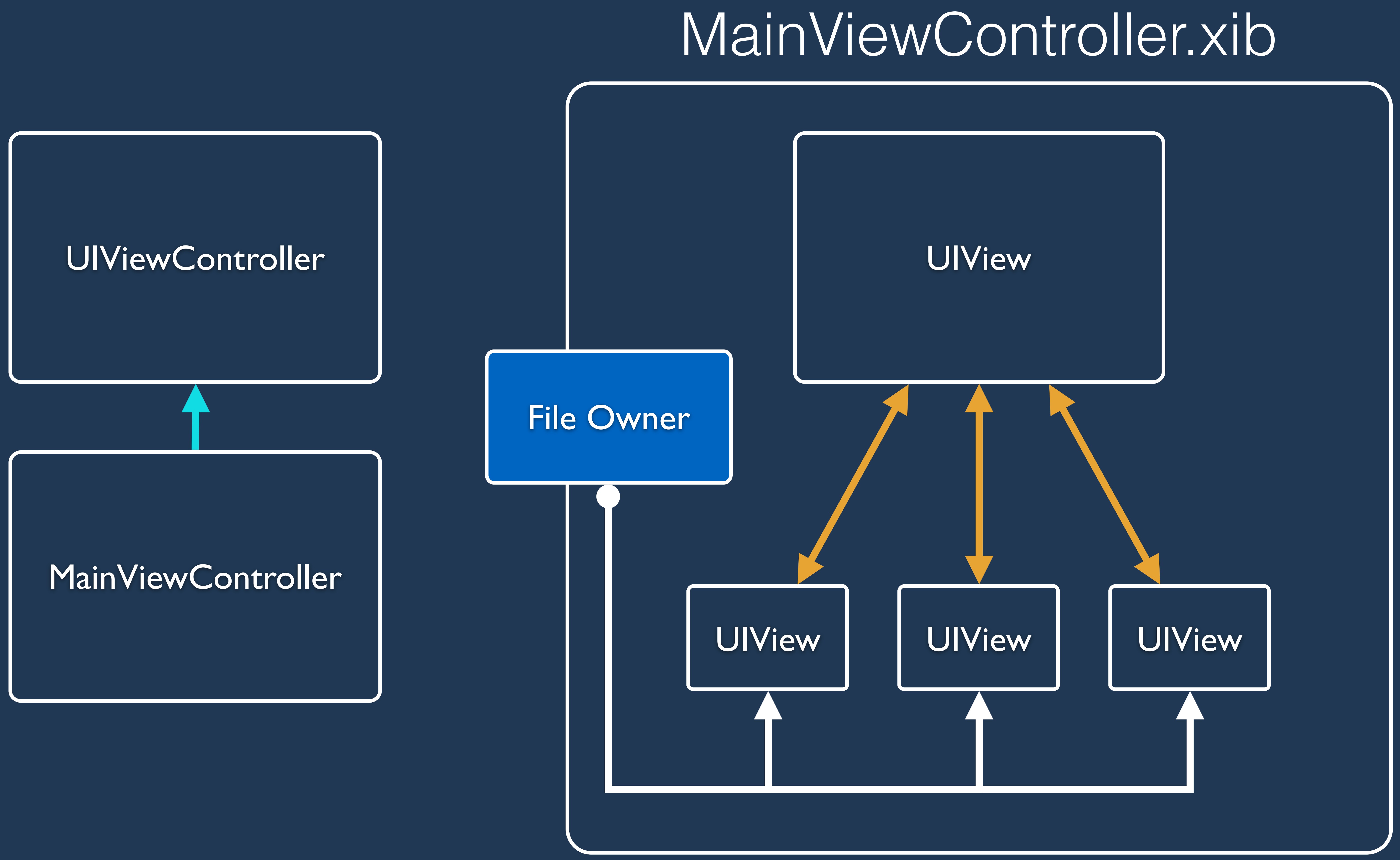
```
MainViewController *main = [[UIViewController alloc]
                              initWithNibName:@“MainViewController”
                                       bundle:nil];
```

.xib loading

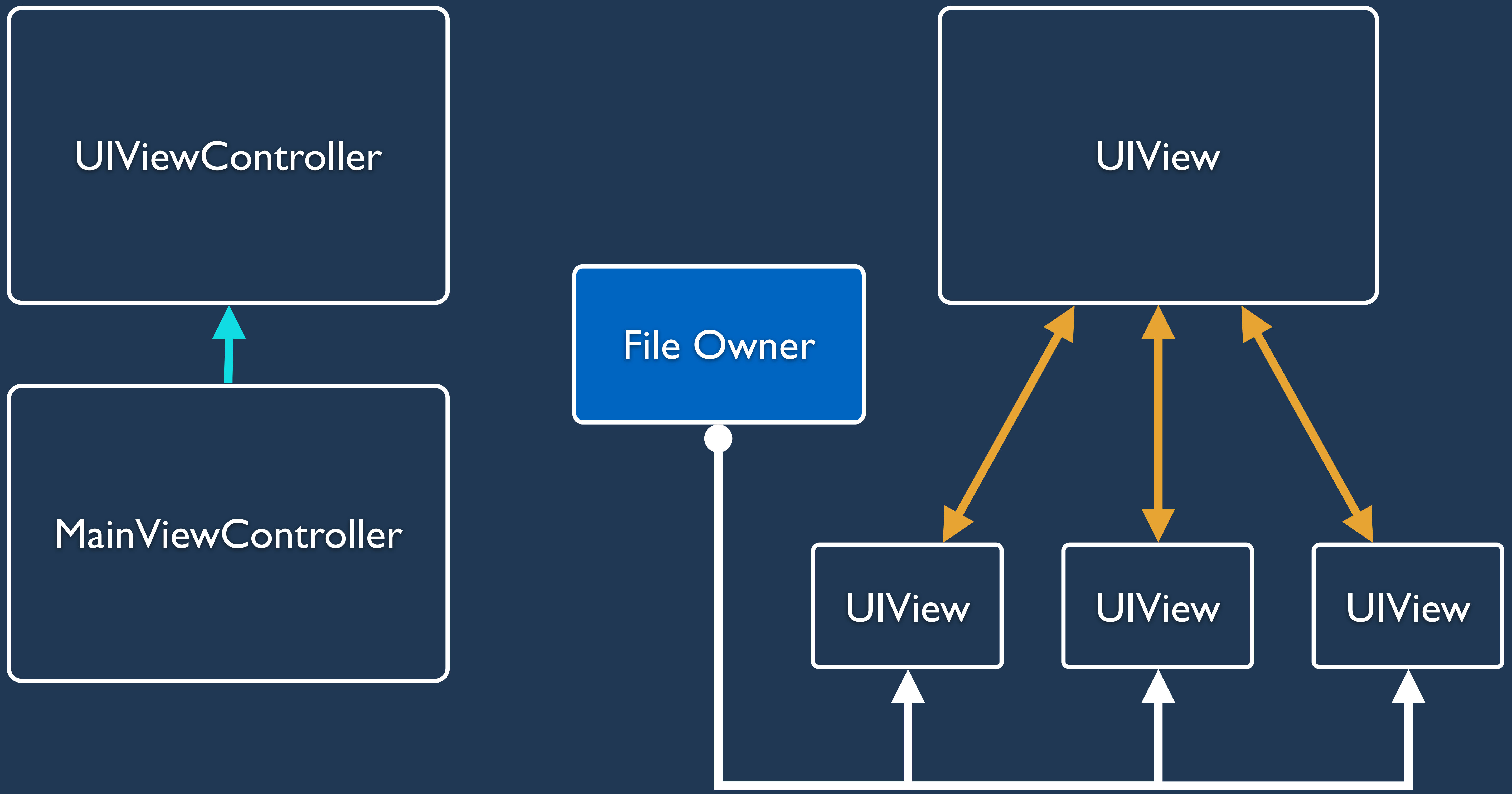


MainViewController.xib
UIViewController
MainViewController
File Owner
UIView
UIView
UIView
UIView

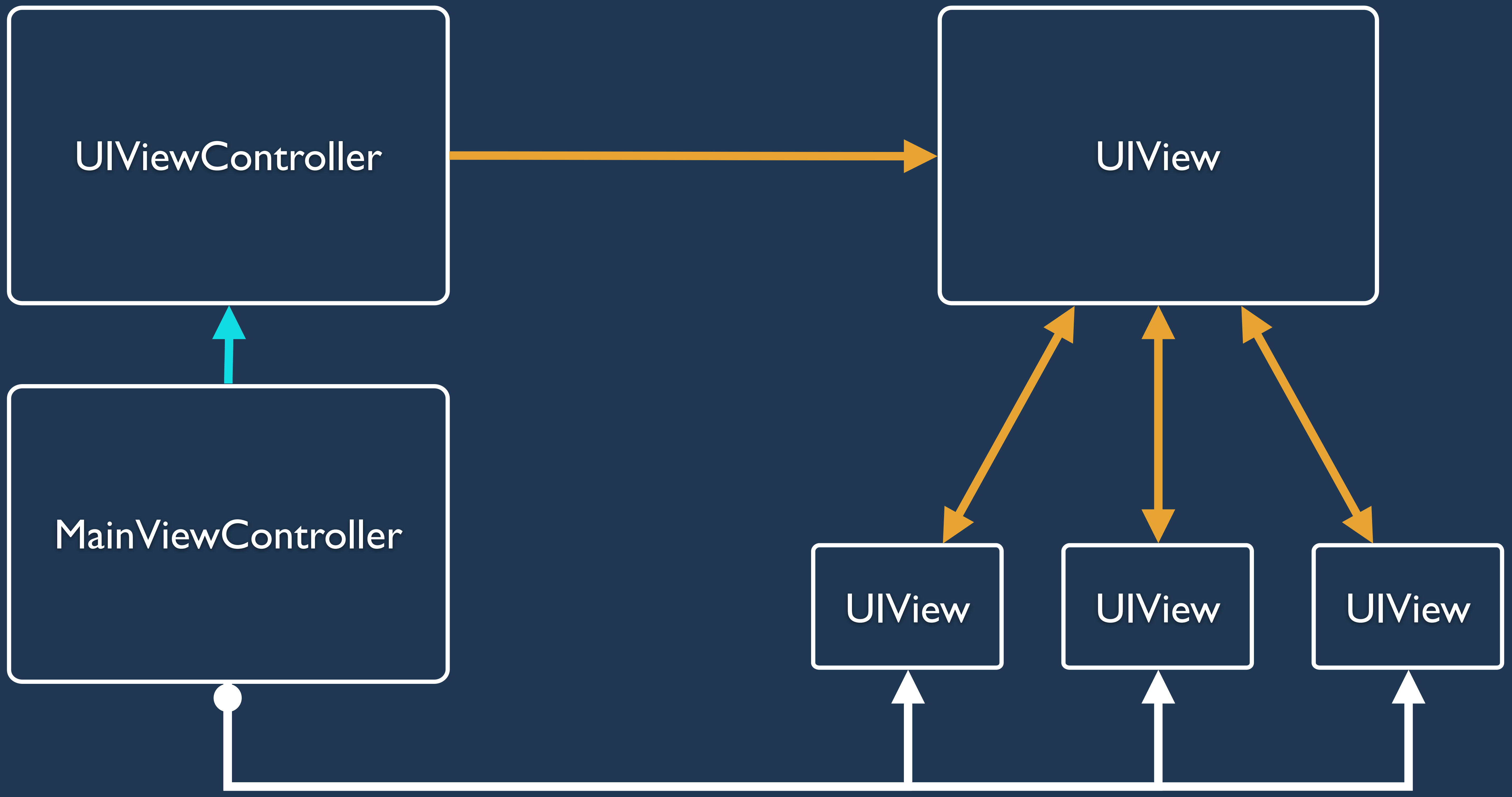
.xib loading


MainViewController.xib
UIViewController
MainViewController
File Owner
UIView
UIView
UIView
UIView

.xib loading


UIViewController
MainViewController
File Owner
UIView
UIView
UIView
UIView

.xib loading


UIViewController
UIView
MainViewController
UIView
UIView
UIView

Outlets

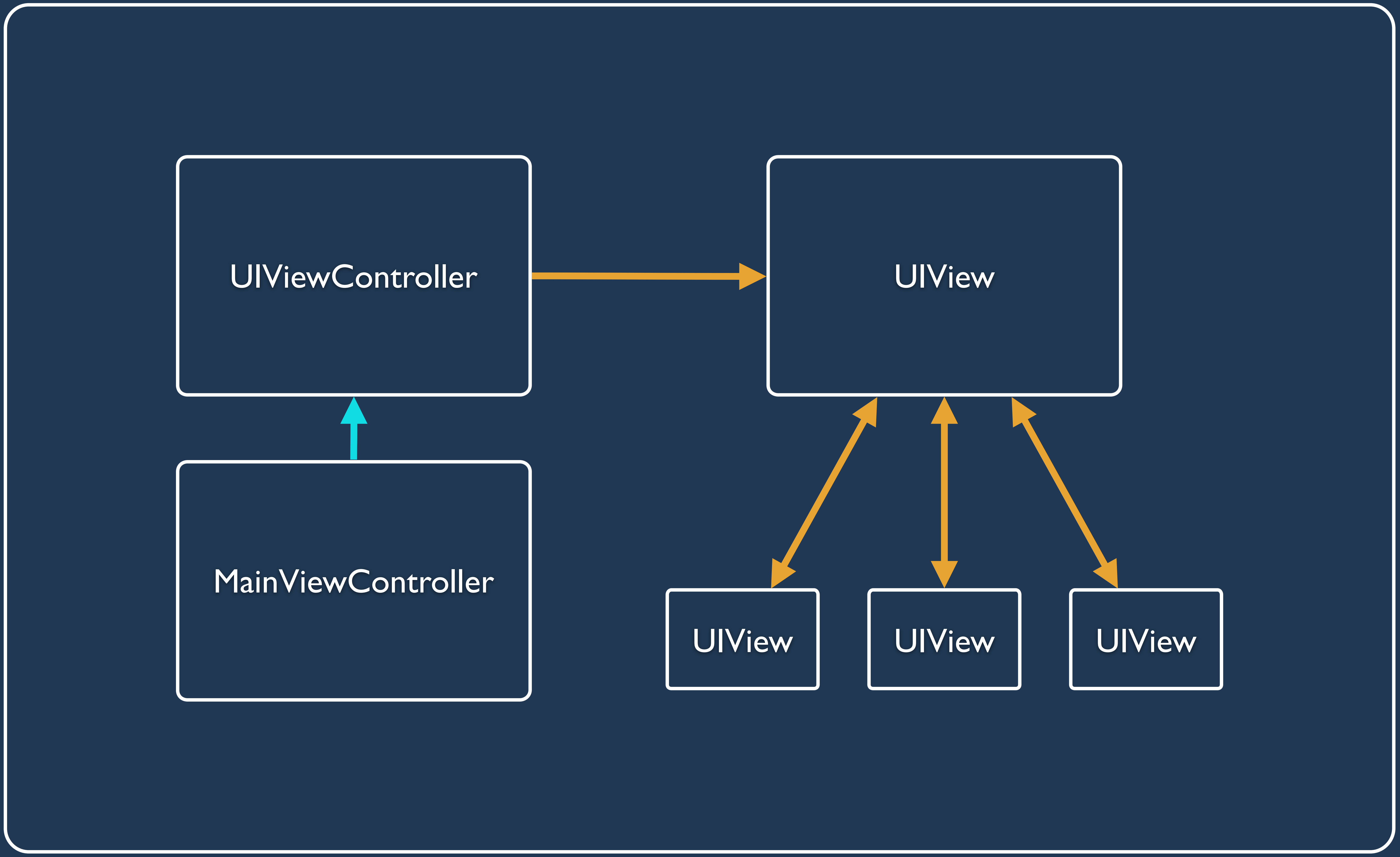
```
@interface MainViewController ()

@property (weak, nonatomic) IBOutlet UIView *articlesView;

@end
```

Storyboard file

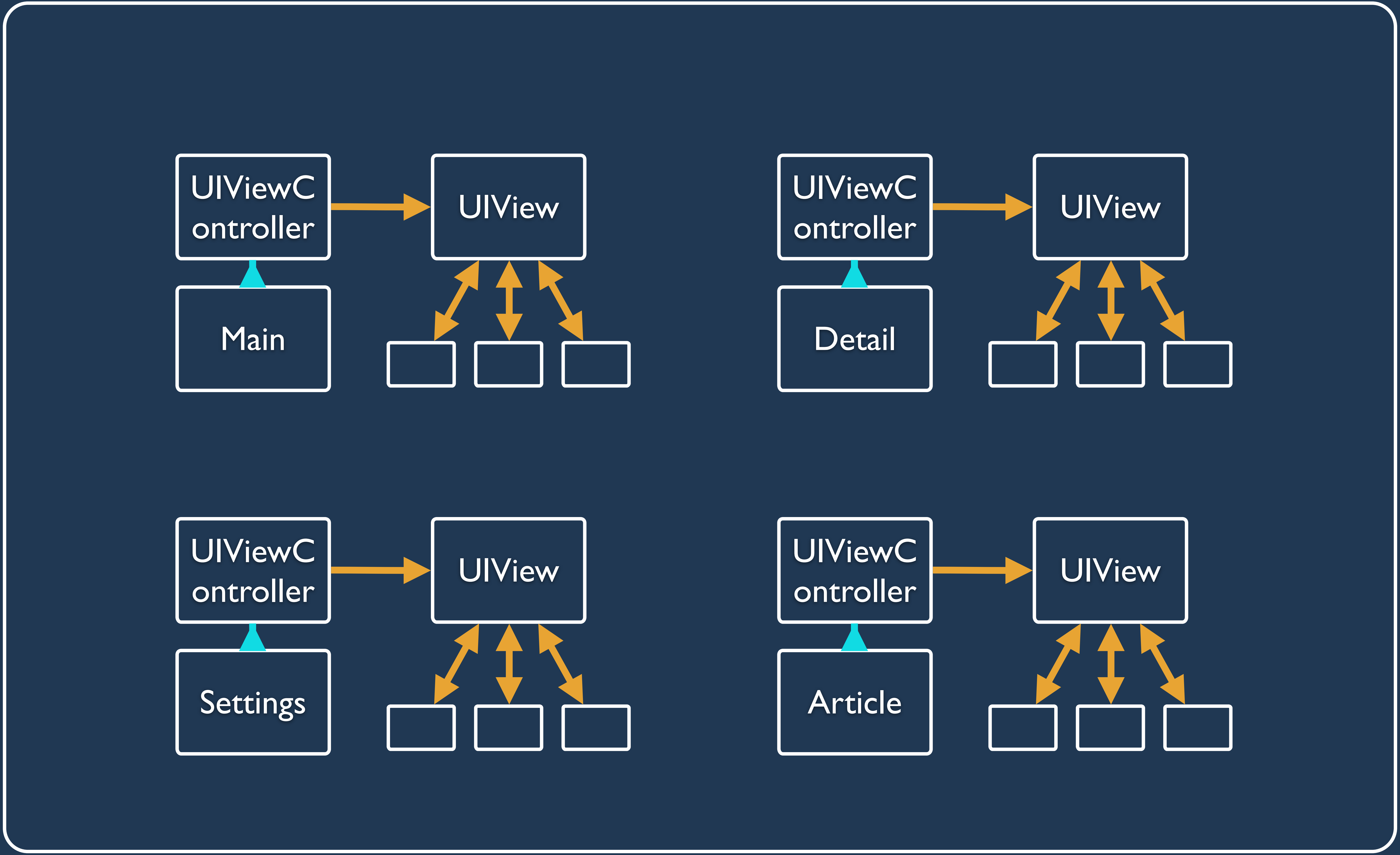
Storyboard

- Just like a .xib, but can contain multiple view controllers
- Defines connections, or segues, between view controllers
- Instantiates view controllers on your behalf

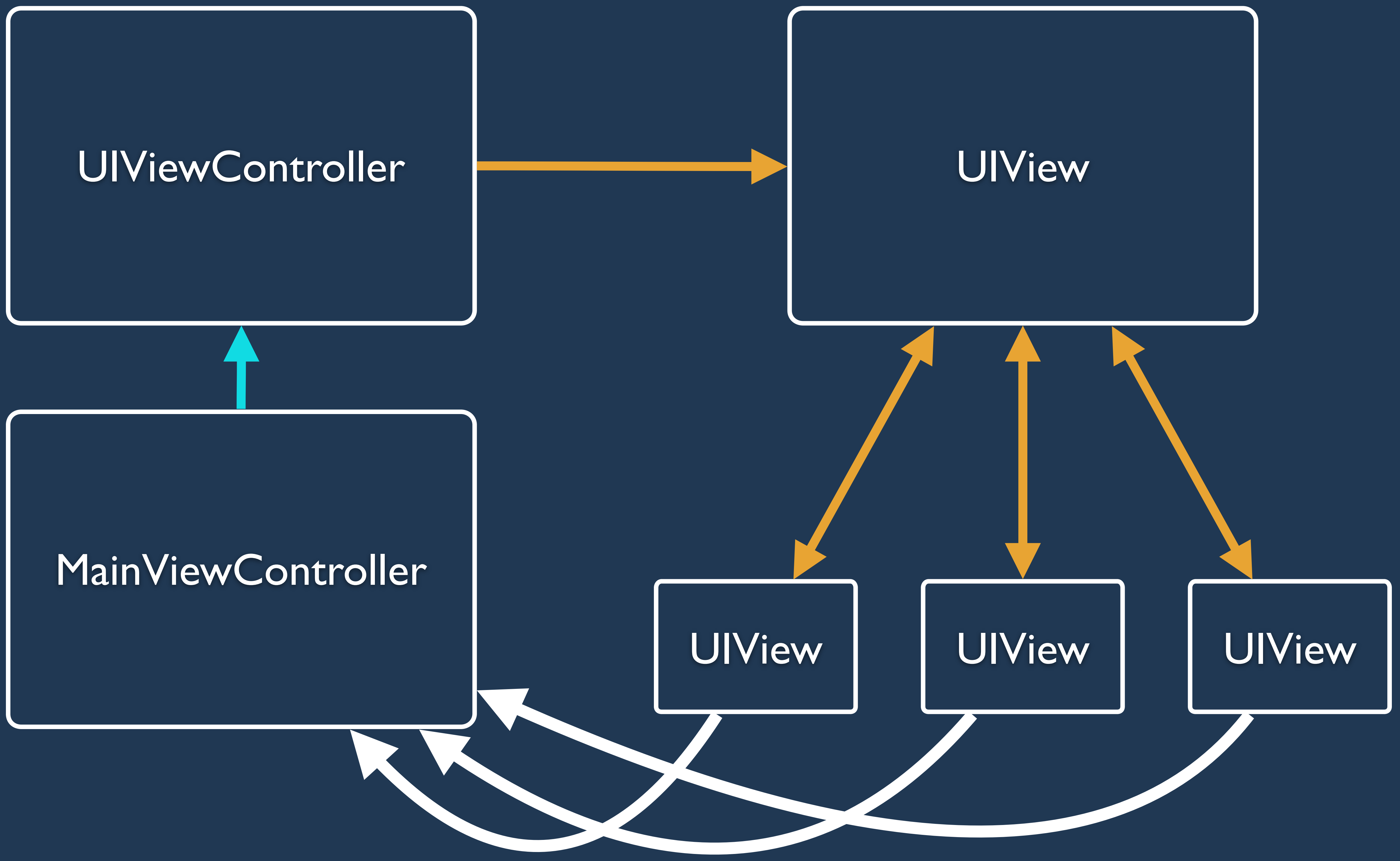
Storyboard


UIViewController
UIView
MainViewController
UIView
UIView
UIView

Storyboard


UIViewController
UIView
Main
UIViewController
UIView
Detail
UIViewController
UIView
Settings
UIViewController
UIView
Article

Responding to events

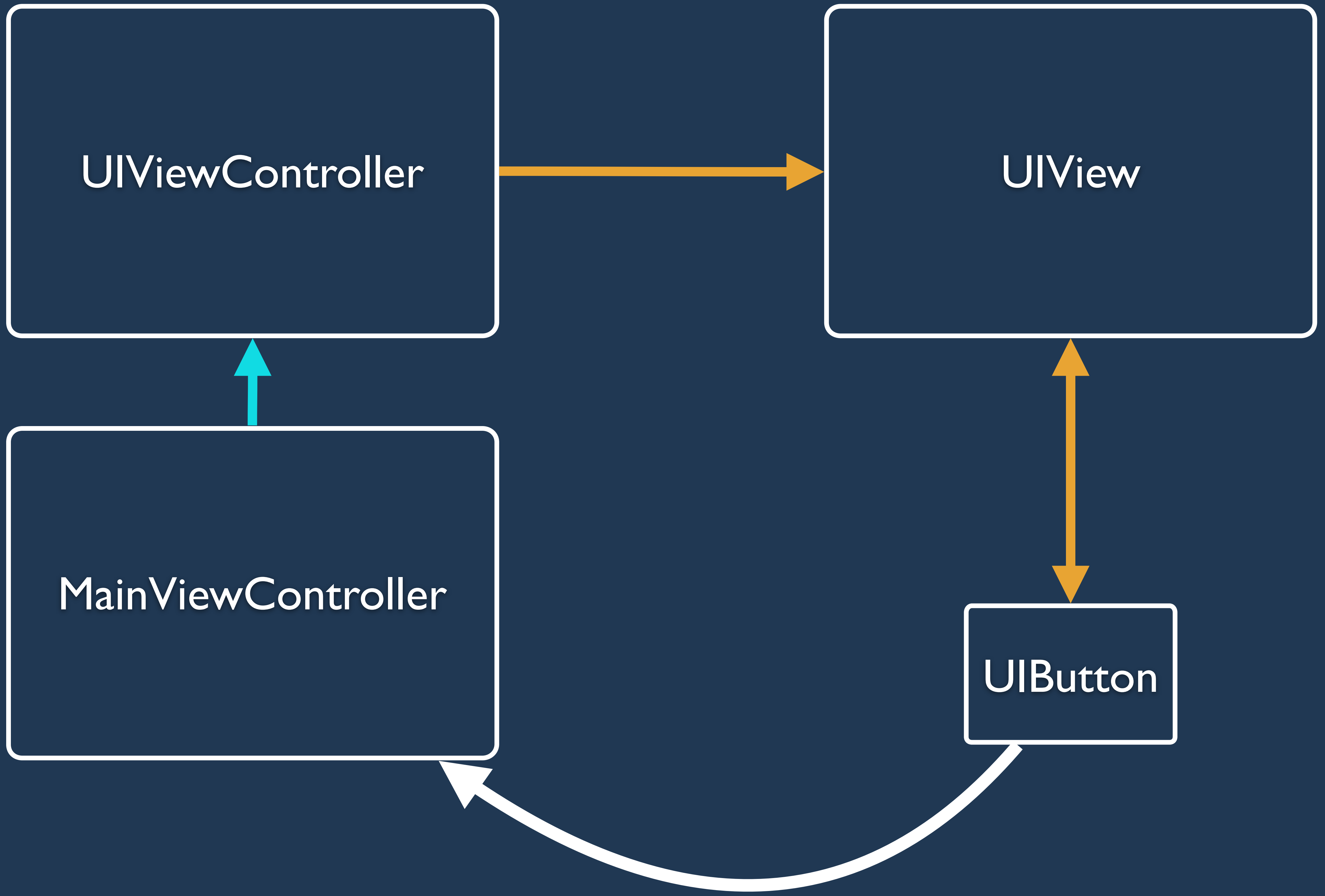


Target-Action

Target-Action

- Provides a structured way for one object (often a UIView) to communicate with another object (often a UIViewController).
- A target - any NSObject
- An action - a message to be sent when an event is triggered

Responding to events



“When this button is tapped, call the showArticles: method”

Target-Action

```
@interface MainViewController ()

- (IBAction)showArticles:(id)sender;

@end
```

Target-Action

- Connect the action via Interface Builder, much like an outlet
- Or call addTarget:action:forControlEvents: in code
- Views can trigger multiple targets with multiple actions

Lab 2.2