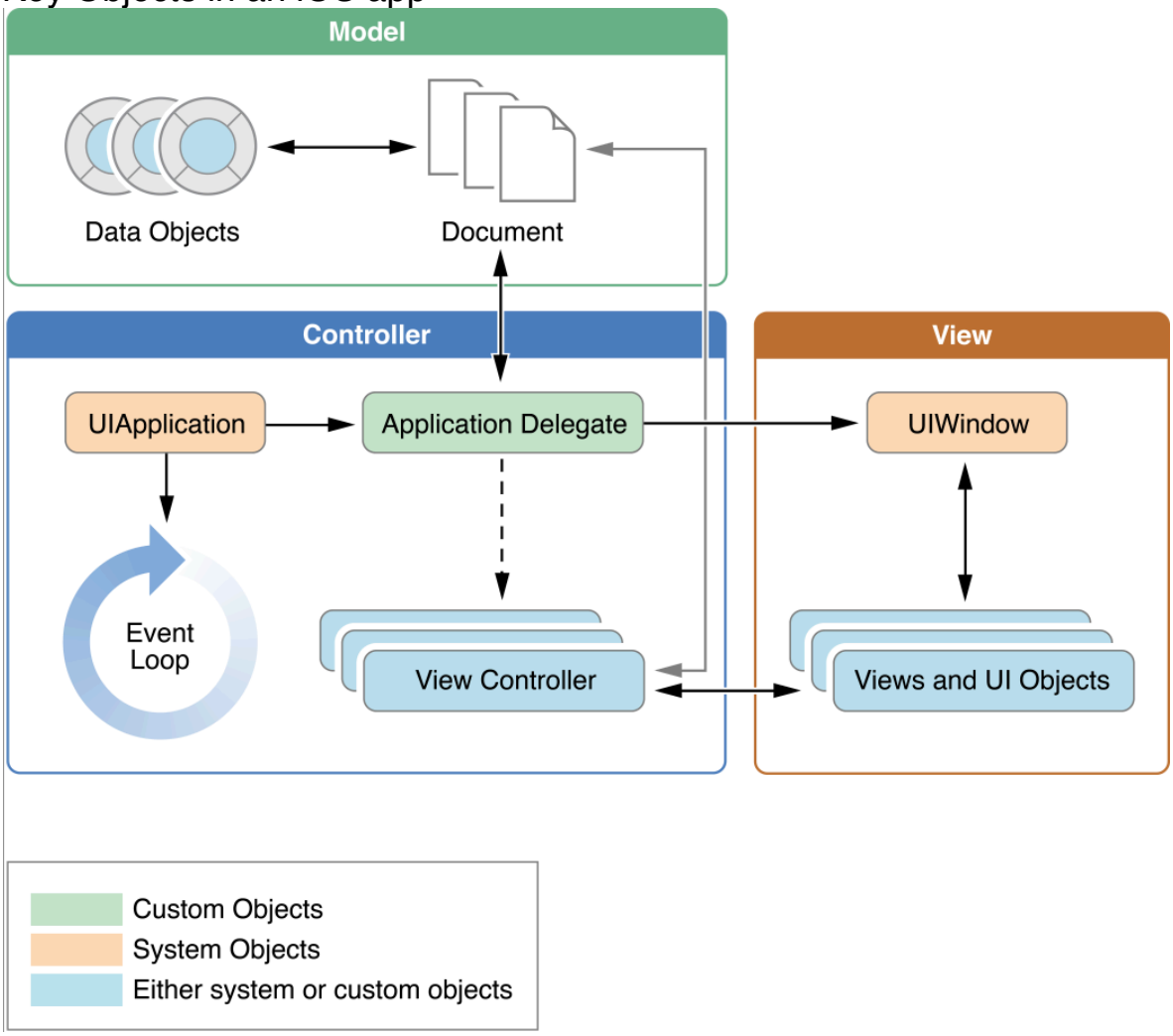
Know the Core Objects of Your App

Key Objects in an iOS app



Role of Objects in an iOS App

> UIApplication object

manage the app event loop and coordinates other high-level app behaviors.

> App delegate object

the app delegate is a custom object created at app lunch time, usually by UIApplicationMain function.

the job of this object is to handle state transition within the app.

> Documents and data model objects

data model objects store your app's content and specific to your app.

Apps can use documents objects(custom subclass of UIDocument) to manage some or all of their data model objects.

> View controller objects
it manages the presentation of your app's content on screen.
it manages a single view and its collection of subveiws.

> UIWindow object
it manages the presentation of one or more views on a screen.
Most apps have only one window, which presents content on the main screen, but apps may have an additional window for content displayed on an external display.
In addition to hosting views, windows work with the UIApplication object to deliver events to your views and view controllers.

> View, Control and layer objects
A view is an object that draws content in a designated rectangular area and responds to events within that area.
Controls are a specialized type of view responsible for implementing familiar interface objects such as buttons, text fields, and toggle switches.
Layer objects are actually data objects that represent visual content. Views use layer objects extensively behind the scenes to render their content. You can also add custom layer objects to your interface to implement complex animations and other types of sophisticated visual effects.

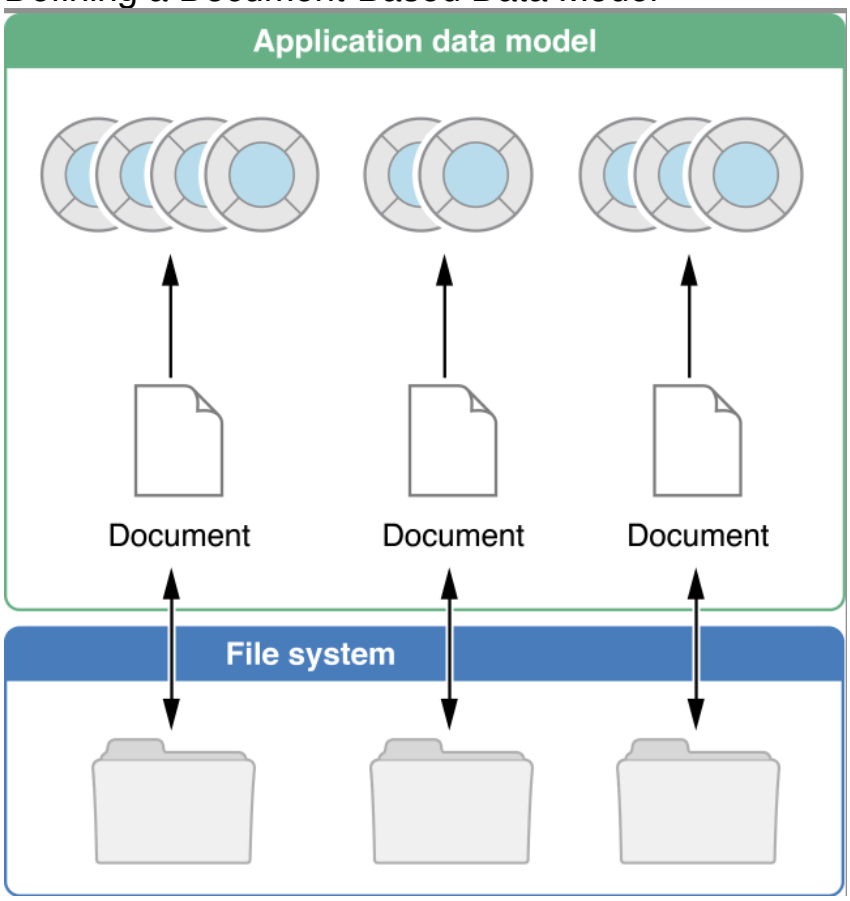
Data Model
Define a custom data model
Data classes in the Foundation framework (data-related objects)
> NSString , NSMutableString, NSAttributedString, NSMutableAttributedString(Strings and Text)
> NSNumber, NSDecimalNumber, NSIndexPath(Numbers)
> NSData, NSMutableData, NSValue(Raw bytes)
> NSDate, NSDateComponents(Dates and Times)
> NSURL(URLs)
> NSArray, NSMutableArray, NSDictionary, NSMutableDictionary, NSIndexSet, NSMutableIndexSet, NSOrderedSet, NSMutableOrderedSet, NSSet, NSMutableSet(Collections)

Data types
> NSInteger, NSUInteger
> NSRange
> NSTimeInterval

> CGPoint
> CGSize
> CGRect

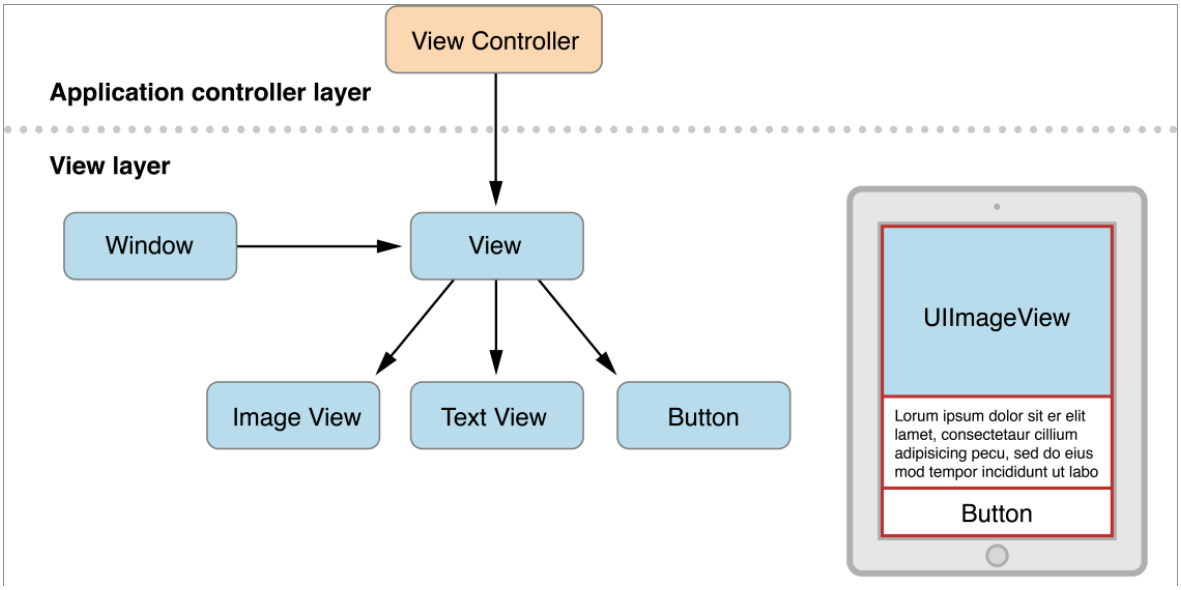
Defining a Structured Data Model Using Core Data

Defining a Document-Based Data Model

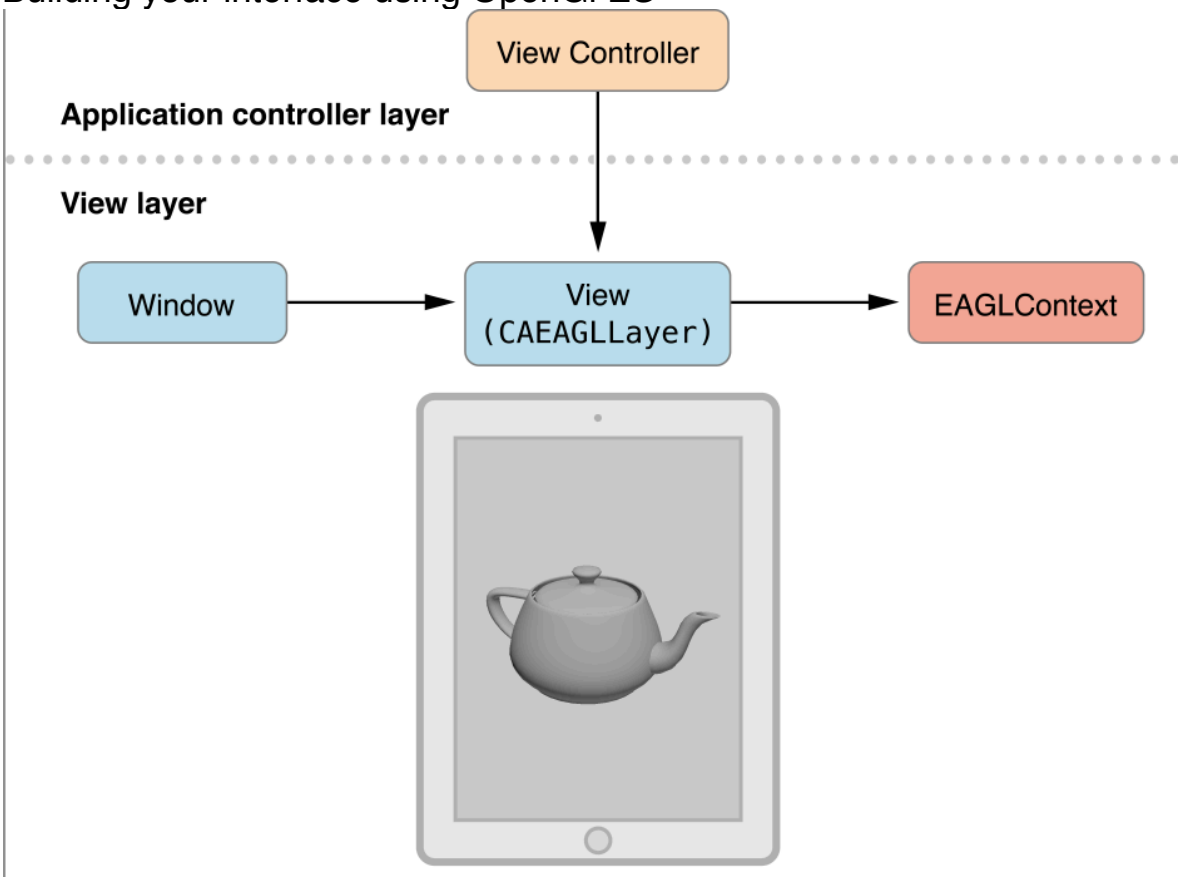


Building an Interface Using UIKit Views
the advantage of interfaces based on UIKit views is that you can assemble them graphically using interface builder.

Building your interface using view objects



Building an Interface Using Views and OpenGL ES
Building your interface using OpenGl ES



the OpenGL ES view is backed by a different type of layer object (a CAEAGLLayer object) instead of the standard layer used for view-based apps.
The CAEAGLLayer object provides the drawing surface that OpenGL ES can render into.
To manage the drawing environment, the app also creates an EAGLContext object and stores that object with the view to make it easy

to retrieve.

The App Bundle
a bundle is a directory in the file system that groups related resources in one place.