**App Life Cycle**

* All Apps are sophisticated (having deal complexity)between your custom code and the system framework.
* Means they are in between your code and frameworks.
* The system framework provide the basic infrastructure that all apps need to run.
* iOS uses most important design pattern such as model -view -controller delegation in their implementation.
* The Model-View-Controller (MVC) it is design pattern assigns objects in an application one of three roles: model, view, or controller.
* **Model**: Represents the business logic of your application.
* **View**: Represents what the user sees in the device.
* **Controller:** Acts as a mediator between the Model and View.
* There should not be any direct conversation between the View and the Model.
* The Controller updates the View based on any changes in the underlying Model.
* If the user enters or updates any information in the View, the changes are reflected in the Model with the help of the Controller.
* The pattern defines not only the roles objects play in the application, it defines the way objects communicate with each other.
* Each of the three types of objects is separated from the others by abstract boundaries and communicates with objects of the other types across those boundaries.

**Main Function:**

* Every App starting point is the main function in C.
* In iOS Apps you not write main by yourself. Instead of XCode only creates this function for basic project.

**The main function of iOS App :**

#import <UIKit/UIKit.h>

#import "AppDelegate.h"

int main( int argc, char \* argv[])

{

@autoreleasepool

{

return UIApplicationMain(argc, argv, nil NSStringFromClass ( [ AppDelegate Class ]))

}

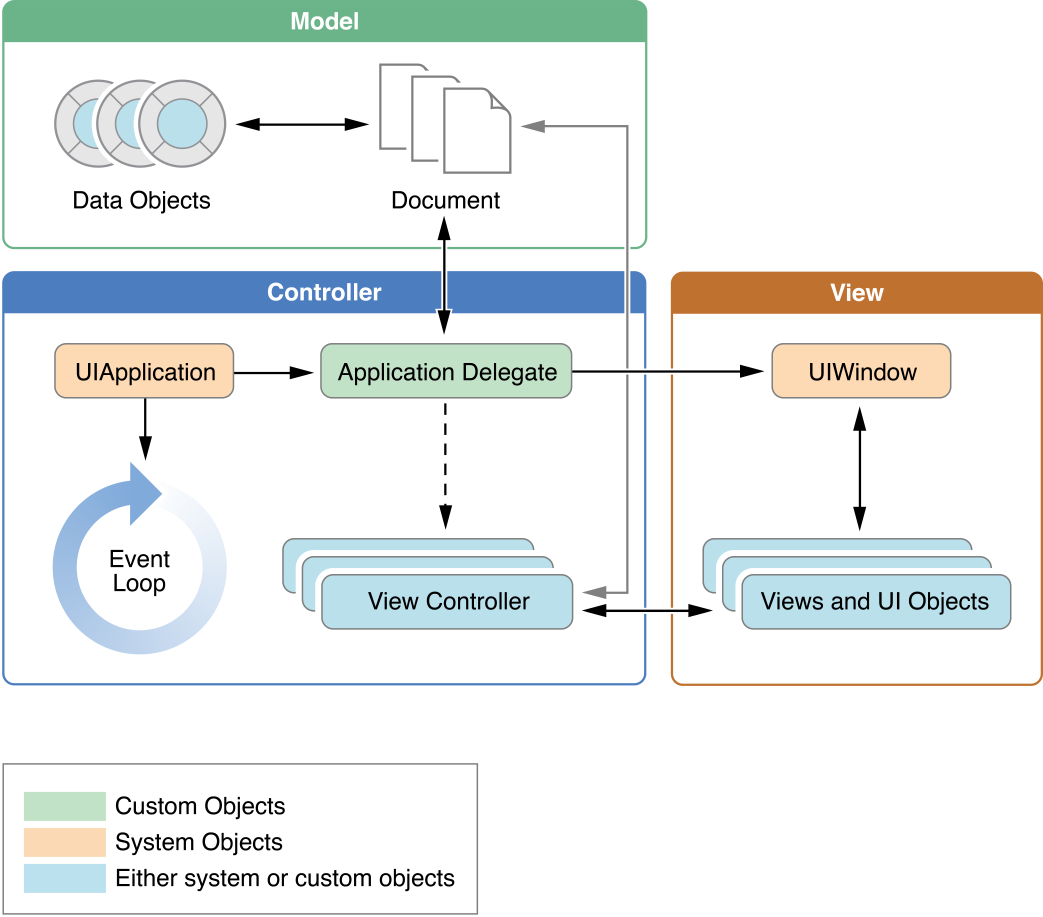
}

* The [UIApplicationMain](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKitFunctionReference/index.html#//apple_ref/c/func/UIApplicationMain) function handles this process by creating the core objects of your app, loading your app’s user interface from the available storyboard files, calling your custom code so that you have a chance to do some initial setup, and putting the app’s run loop in motion.

**The Structure Of an App:**

* From Starting point [UIApplicationMain](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKitFunctionReference/index.html#//apple_ref/c/func/UIApplicationMain)  function is only sets the keyObjects and app will be start running.
* At the heart of every iOS app is the [UIApplication](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplication_Class/index.html#//apple_ref/occ/cl/UIApplication) object, whose job is to facilitate the interactions between the system and other objects in the app.

**The key Objects in iOS App :**



**Fig: shows MVC Design Pattern**

**The Main Roles Of Objects in iOS App**

**i)UIApplication Object:**

* The UIApplication class provides a centralized point of control and coordination for apps running in iOS.
* When an app is launched, the system calls the [UIApplicationMain](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKitFunctionReference/index.html#//apple_ref/c/func/UIApplicationMain) function.
* The UIApplication Object it manages event loop behaviour.

**ii)App delegate Object:**

**Delegation:**

* It is a simple and powerful pattern in which one object in a program acts on behalf of, or in coordination with, another object. It act like protocol.
* The UIApplicationDelegate protocol defines methods that are called by the singleton [UIApplication](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplication_Class/index.html#//apple_ref/occ/cl/UIApplication) object in response to important events in the lifetime of your app..
* The App delegate is heart of your code .
* This Object mainly work with UIApplication Object to handle app initalization, state transistion.

**iii)Data Model Objects:**

* It is used to store app contents.
* Example: Banking app it store customer Financial Transaction in database.

**Document Objects:**

* Apps also use to manage all their data model Objects.

**iv)View Controller Objects:**

* View Controller is a class it is base class for all.
* View controllers are the foundation of your app's internal structure. Every app has at least one view controller, and most apps have several. Each view controller manages a portion of your app's user interface as well as the interactions between that interface and the underlying data.
* It used for presentation of your app Contents on display screen.
* View controllers  is class it is base class for all other View Controller Objects.
* The  functionality for loading views, presenting them, rotating them in response to device rotations .
* UIKit and other frameworks define additional view controller classes to implement standard system interfaces such as the image picker, tab bar interface, and navigation interface.

v) **UIWindow Object**:

* This Object it co ordinates the presentation on One or more view in display screen.
* Many Apps have only one window means content represents on the main screen.
* but apps may have an additional window for content displayed on an external display.
* You want to change the content of your app, you use a view controller to change the views displayed in the corresponding window. You never replace the window itself.

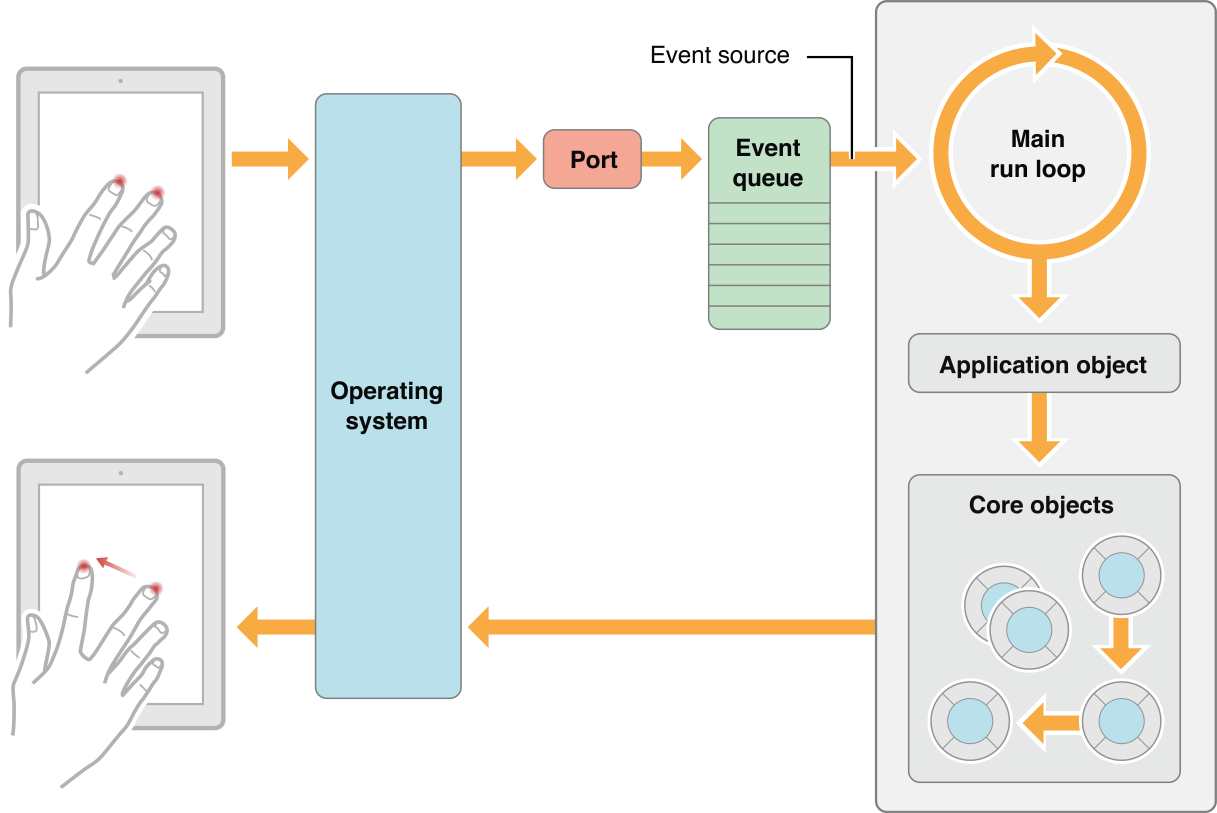
**View Objects, Conrol Objects and Layer Objects**:

* **View**:  A view is an object that draws content in a designated rectangular area and responds to events within that area.
* **Control Object**: It is specialized type of view it is responsible for implementing familiar interface objects such as buttons, text fields, and toggle switches.
* **Layer Objects**: Layer objects are actually data objects that represent visual content. Views use layer objects intensively behind the scenes to render their content

**The Main Run Loop:**

* The [UIApplication](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplication_Class/index.html#//apple_ref/occ/cl/UIApplication) object sets up the main run loop at launch time and uses it to process events and handle updates to view-based interfaces
* The main run loop executes on the app's main thread
* Run loops are part of the fundamental infrastructure associated with threads.
* A run loop is an event processing loop that you use to schedule work and coordinate the receipt of incoming events.
* The purpose of a run loop is to keep your thread busy when there is work to do and put your thread to sleep when there is none.

**Architecture Of Main Run Loop:**



**Fig: shows The Process Of an Application Running Architecture**

* As the user interacts with a device, events related to those interactions are generated by the system and delivered to the app via a special port set up by UIKit.
* Events are queued internally by the app and dispatched one-by-one to the main run loop for execution.
* The UIApplication object is the first object to receive the event and make the decision about what needs to be done. A touch event is usually dispatched to the main window object, which in turn dispatches it to the view in which the touch occurred.

**Events For iOS Apps:**

|  |  |  |
| --- | --- | --- |
| **Event type** | **Delivered to…** | **Notes** |
| Touch | The view object in which the event occurred. | The Views are responder objects.  Any touch events which are not handled by the view are forwarded down the responder chain for processing. |
| Remote control  Shake motion events | First responder object. | The Remote control events are like controlling media playback which are generated by headphones and other accessories. |
| Accelerometer  Magnetometer  Gyroscope | The object we define. | Events related to the accelerometer, magnetometer, and gyroscope hardware are delivered to the object we define. |
| Location | The object we define for location | We register to receive location events using the Core Location framework. |
| Redraw | The view that needs the update | The Redraw events do not involve an event object but are simply calls to the view to draw itself. |

**App Execution States :**

* **Means The App's in which state running How the execution while before or after Using your app.**

**Example**:

* when the user presses the Home button, a phone call comes in, or any of several other interruptions occurs, the currently running apps change state in response.

**Block Diagram For Apple Life Cycle Different States**

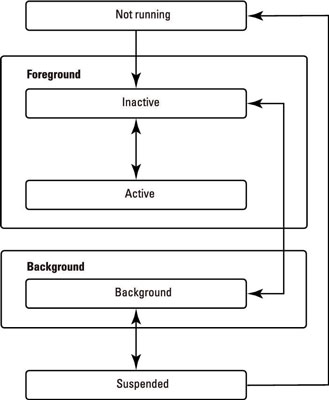


Fig: Different States Of Apple Life Cycle

**Different States**

* **Not Running:**
* Means The app has not been launched or was running but was terminated by the system.
* **In Active:**
* The App is running in the foreground but it is currently not receving any events. Foreground means(the part of a view that is nearest to the observer).
* **Active:**
* The app is running in foreground and it also receiving events
* The foreground contains the applications the user is working on,
* **Background:**
* background contains the applications that are behind the scenes.
* The app is in the background and executing the code.
* When the user is not actively using your app, the system moves it to the background state. For many apps, the background state is just a brief stop on the way to the app being suspended. Suspending apps is a way of improving battery life .
* **Suspended:** The app is in the background but it is not executing the code.
* While suspended, an app remains in memory but does not execute any code
* Most state transitions are done through by a corresponding call to the methods of your app delegate object.

**Methods in Application Life Cycle**

1. [**Application:willFinishLaunchingWithOptions:**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/application:willFinishLaunchingWithOptions:) This method says your app’s first chance to execute code at launch time.
2. [**Application:didFinishLaunchingWithOptions:**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/application:didFinishLaunchingWithOptions:) This method says you to perform any final initialization before your app is displayed to the user.
3. [**ApplicationDidBecomeActive:**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/applicationDidBecomeActive:)This Method says that it is about to become the foreground app. Use this method for any last minute preparation.
4. **A**[**applicationWillResignActive:**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/applicationWillResignActive:) This method says your app is transitioning away from being the foreground app. Use this method to put your app into a quiescent state.
5. [**ApplicationDidEnterBackground:**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/applicationDidEnterBackground:)—This method says your app is now running in the background and may be suspended at any time.
6. [**ApplicationWillEnterForeground:**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/applicationWillEnterForeground:)This method says that your app is moving out of the background and back into the foreground, but that it is not yet active.

7.[**ApplicationWillTerminate**:](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/applicationWillTerminate:)This method says that your app is being terminated. This method is not called if your app is suspended.

**App Termination**:

* Apps terminated Suspended apps receive no notification.
* The system usually terminates apps so that it can reclaim memory and make room for other apps being launched by the user, but the system may also terminate apps that are misbehaving or not responding to events in a timely manner.
* when they are terminated at any time and should not wait to save user data or perform other critical tasks.
* If an app is currently running in the background and not suspended, the system calls the [applicationWillTerminate:](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html#//apple_ref/occ/intfm/UIApplicationDelegate/applicationWillTerminate:) of its app delegate prior to termination.

**Thread and Concurrency:**

* The system creates your app’s main thread and you can create additional threads, If you want to perform other tasks.
* For iOS apps, the preferred technique is to use Grand Central Dispatch (GCD), operation objects, and other asynchronous programming interfaces rather than creating and managing threads yourself.

Threads and Concurrency some Operation:

* Work involving views, Core Animation, and many other UIKit classes usually must occur on the app’s main thread. There are some exceptions to this rule—for example, image-based manipulations can often occur on background threads—but when in doubt, assume that work needs to happen on the main thread.
* Lengthy tasks (or potentially length tasks) should always be performed on a background thread. Any tasks involving network access, file access, or large amounts of data processing should all be performed asynchronously using GCD or operation objects.
* At launch time, move tasks off the main thread whenever possible. At launch time, your app should use the available time to set up its user interface as quickly as possible. Only tasks that contribute to setting up the user interface should be performed on the main thread. All other tasks should be executed asynchronously, with the results displayed to the user as soon as they are ready.