**App life cycle**

* Apps are a sophisticated interplay between your custom code and the system frameworks.
* The system frameworks gives the infrastructure that all apps need to run to get the required features we want.
* The frameworks depends on design patterns such as model-view-controller.

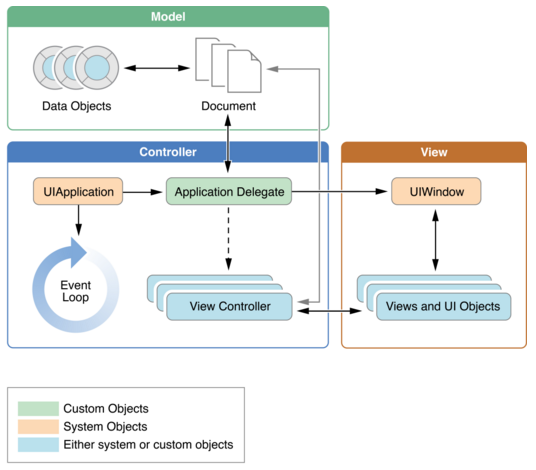
**Model-View-Controller:**

* It is a design patter that assigns the objects in application one of the role: model, view or controller.
* The pattern defines not only the role but also the communication between two objects.
* It is good design for cocoa application.
* The application built on MVC design are more extensible than other application.

**Main function:**

* Always the execution starts from main method and its main function is to hand control off to the UIKit framework.
* Xcode generates a main method for the iOS application.
* The only function of main function is to handover the control to the UIKit.
* 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.

**Structure of an App:**



* During startup, the [UIApplicationMain](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKitFunctionReference/index.html#//apple_ref/c/func/UIApplicationMain) function sets up several key objects and starts the app 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.
* iOS app uses the MVC model.
* This architecture is crucial to creating apps that can run on different devices with different screen sizes.

**Role of objects in an iOS app:**

1] [UIApplication](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplication_Class/index.html#//apple_ref/occ/cl/UIApplication)object: It manages the event loop and other high-level app behaviors.

2] App delegate object: works in tandem with the UIApplication object to handle app initialization, state transitions, and many high-level app events.

3] Documents and data model objects: Data model object stores app’s content and are specific to your app and Document object manage some or all of their data model objects.

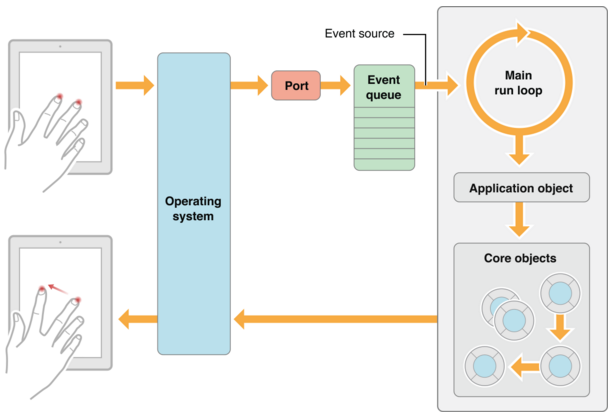
4] View controller objects: It manage the presentation of app’s content on screen.

5] [UIWindow](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIWindow_Class/index.html#//apple_ref/occ/cl/UIWindow) object: This object coordinates the presentation of one or more views on a screen.

6] View objects,control objects, and layer objects: 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.

**The main Run Loop:**

* main run loop processes all user-related events.
* 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.
* This behavior ensures that user-related events are processed serially in the order in which they were received.
* The below figure shows processing events in main run loop.
* 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.

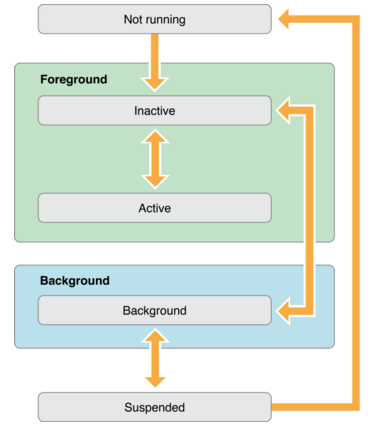


**Common types of events for an iOS App:**

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

**Execution states for App:**

* At any given moment, the app is in one of the following states: Not running, Inactive, Active, Background, Suspended.



1] **Not Running:** The app has not been launched or was running but was terminated by the system.

2] **Inactive:** The app is running in the foreground but is currently not receiving events.

3] **Active:** The app is running in the foreground and is receiving events.

4] **Background:** The app is in the background and executing code. Most apps enter this state briefly on their way to being suspended. However, an app that requests extra execution time may remain in this state for a period of time.

5] **Suspended:** The app is in the background but is not executing code. The system moves apps to this state automatically and does not notify them before doing so. While suspended, an app remains in memory but does not execute any code.

**App Termination:**

* Apps must be ready for termination to happen at any time it should not wait to save data or perform other tasks.
* System-initiated termination is a normal part of an App life cycle.
* The system usually terminates the app which are not responding in a timely manner, so that it can reclaim the memory for other apps.
* The suspended apps do not receive any notification when they are terminated, the system kills the app and reclaims the memory.
* If the app is running in 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.
* The system does not call this method when the device reboots.