CS4092 - Lecture 3

Requirements and Design, App Structure and Views

Modifying an existing app

- Owners are aware of the development process, and know their own app
- Need guidance on how to balance the users' needs with their own goals
- Need assistance with choosing/designing new features in a data driven way

A Framework for Analysis

- The following is an analysis framework that:
    - Identifies user goals, owner goals and helps identify and quantify the relevance of a set of proposed features
- It follows the following pattern:

1. Identify the owners objective for the app
2. Conduct a survey of users to determine what deficiencies they note in the application
3. Cluster users and their feedback
4. Analyse these results to rank features
5. Report on proposed changes to interface for top ranked features. Provide 1 or more potential solutions

Google Analytics

- On iOS, several features can be tracked including:
    - Campaigns, crashes and exceptions and e-commerce interactions
    - Most interesting however is the ability to track:
        - Events on the interface - suitable for tracking low level work such as visibility or use of a particular feature such as a swipe or button
        - Track Screens visited - suitable for drawing a map of how the app is used - what screens and pages are visited
        - Timings - how long is a user staying on a page
- Usually these dimensions need to be analysed as a whole:
    - Is a user interacting with content while dwelling on a page. Is the interaction directed or undirected? etc

Step 1 - Identify the Owner's Goals

- Meet the owner:
- Interview them about their goals
- Make sure these are specific

Step 2 - Identify and Understand Users

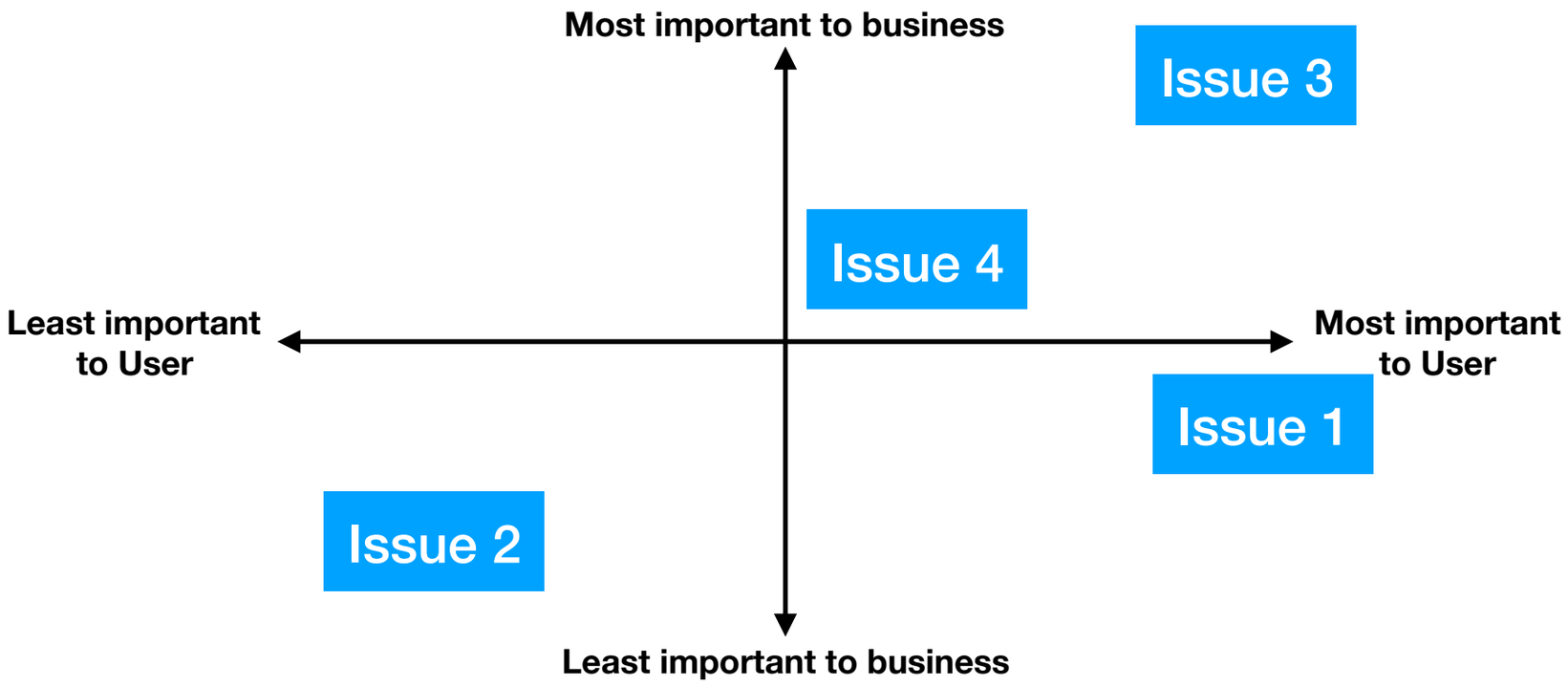
- Understand your users:
    - Include demographic questions in your survey
    - Identify their key behaviours
    - Identify their key needs and goals
- From this information, identify a set of personas
    - These will be centroids when it comes to clustering your users

Step 2 - Identify and Understand Users

- We will now find out how our user uses the current version of the application:
    - Ask the user about their expectations for the application
    - Pose your users (a) task(s) to complete and observe (either through in person or through analytics) how they complete the tasks
    - Survey the user about the task:
        - For instance, how easy, intuitive, satisfaction, etc.

Step 3 - Identify Friction Points

- Cluster your feedback
    - I do this with post-its! Write out each related issue, and group the post-its with a header on a white board.
- Then I post the owner’s goals and the users feedback



Step 4 - Ideate Solutions

- Now we know which ideas are most important, we can take the first one or two and propose solutions
- We will take these to the owner and get their opinion
- If there is sufficient resources, we may prototype and repeat the process with the new solutions
- If not, we implement

Step 4 - Ideate Solutions

- When we send our report:

    1. Describe current process (from users' perspective)
    2. Describe new process
    3. Show interface before and after changes will be made

- We may list two or more solutions - but don't feel obliged. If there is one good solution, remember we are the experts! Choice can complicate things.

Structure of Report

- List owner’s goals
- List clustered features themes identified by the users
    - You would provide a summary of how this list was arrived at
- Show diagram with relative alignment of user feedback and owner goals
- Describe the first N features
    - With potential process followed by user
    - With visuals depicting changes visible to the user

The App Lifecycle

- IOS Apps (and other mobile platforms) are implemented using Design Patterns.

- Two, Model View Controller and Delegate are common throughout an Apps Implementation - often appearing at several layers of the app.

- They allow our code to be structured, and help all parties consider what should occur in the code.

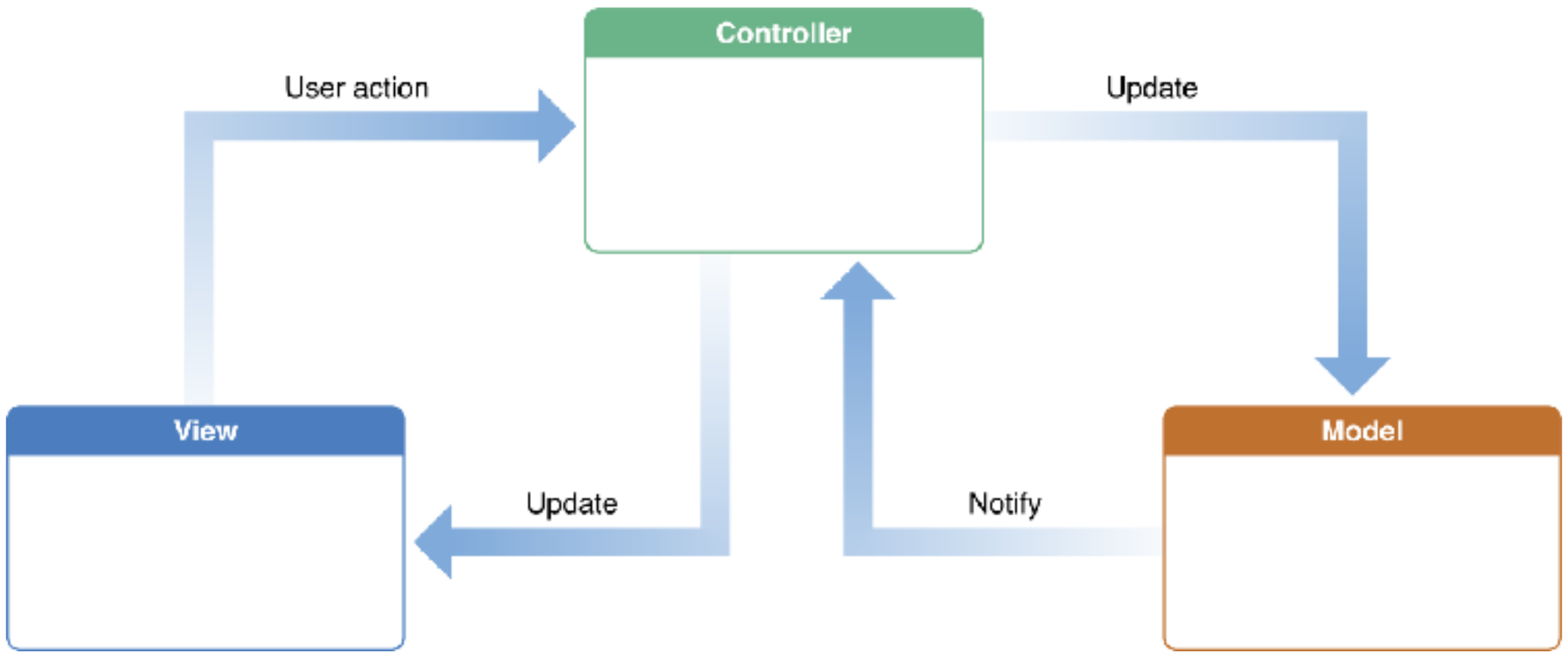
MVC

- Models represent data at some level in the application (either at App Level, or at view level). Models can be updated. Normally this is through the user interface (via the view controller) or through notifications, or even modifications to other model objects.

- View is what the user sees, and uses to interact with the app and its information.

- Controller is an intermediary between the model and the view. It is the logic that determines the consequence of some interaction with the view or update to the model.

MVC



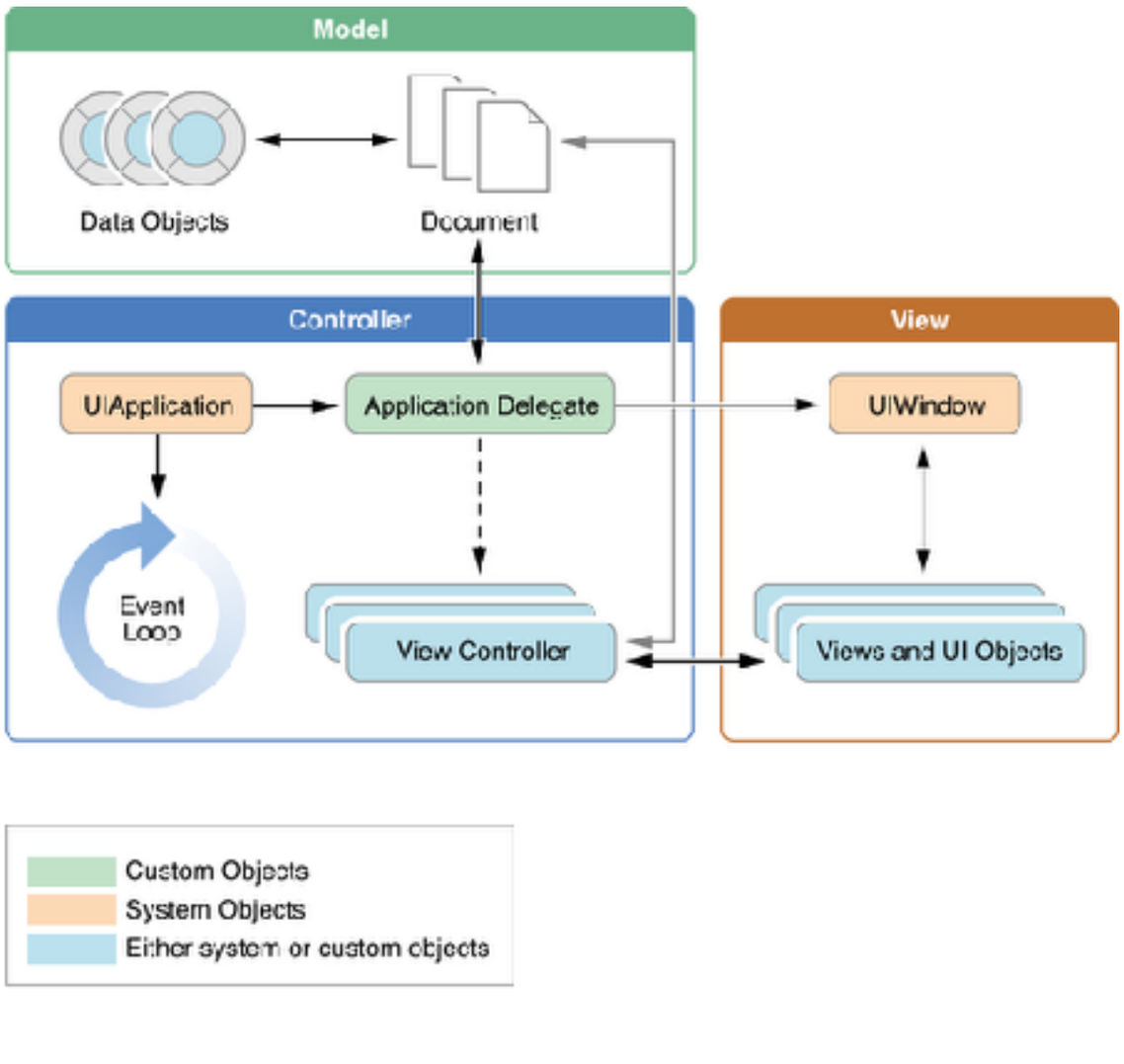
Delegation Pattern

- Delegation is a simple and powerful pattern in which one object in a program acts on behalf of, or in coordination with, another object.

- Typically the delegate implements an interface.

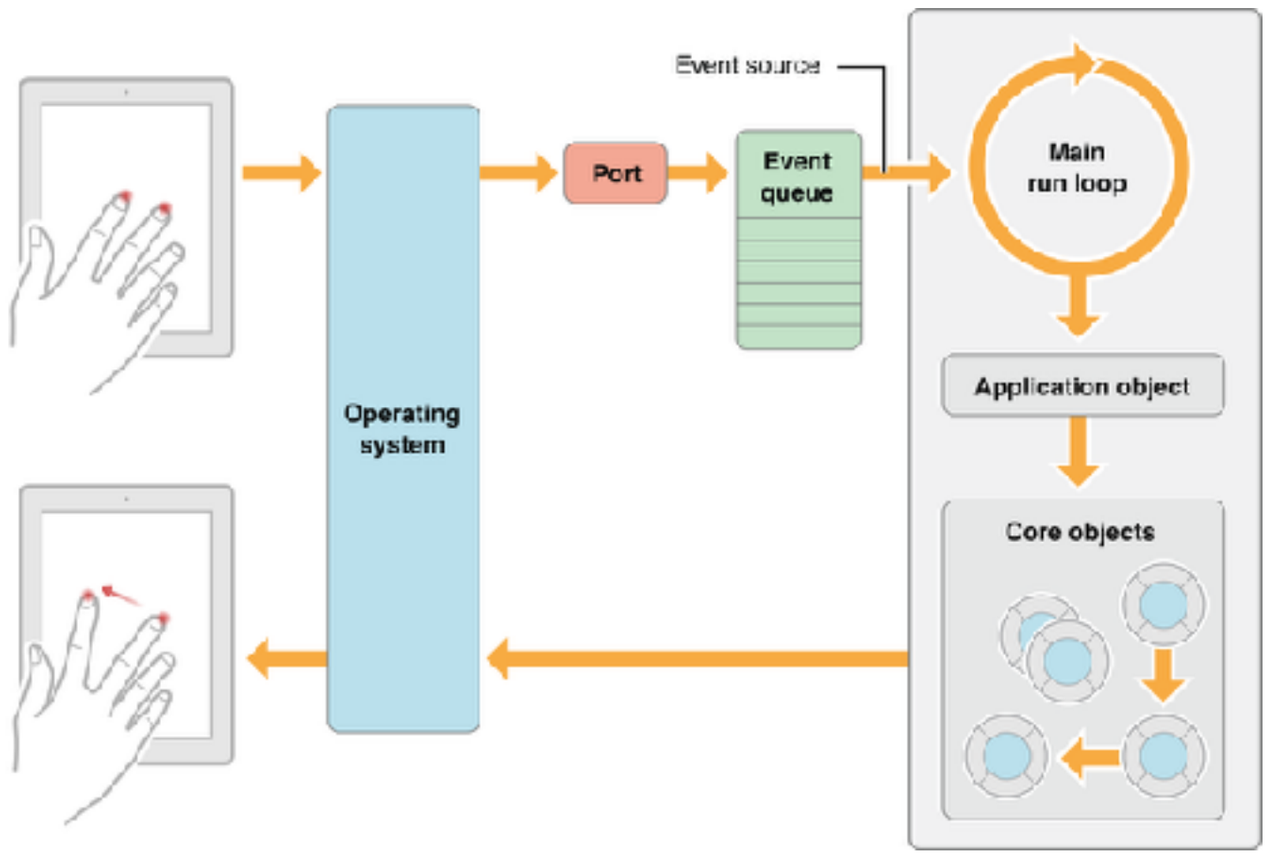
- Admits separation of custom features required by the app from the framework code.

The App Lifecycle

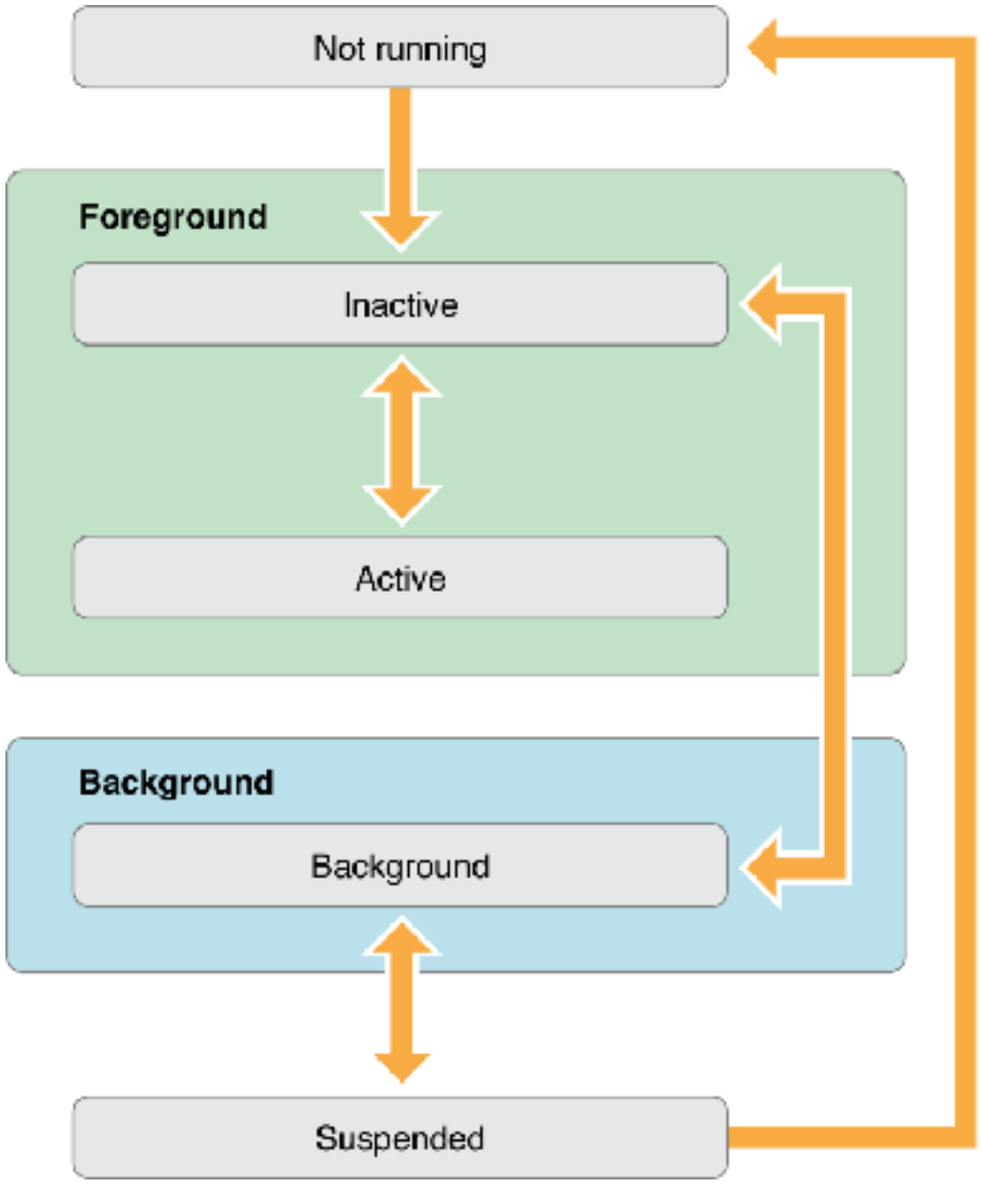


UIApplication	Manages top level operations in app including the event loop
App Delegate	Delegated to by the UIAppication for certain customisable behaviours. Singleton
Data Objects	Data structures to contain our app's data - custom written code as they are unique to each application.
View Controllers	Manage the presentation of our application's screens. They are responsible for handling views' lifecycle, mediating between view and data objects, handling notifications that are relevant to a view and providing logic to determine the consequence of interaction with view controls.
UIWindow	Container for the views.
Views, Controls	Visual representation of components that users can interact with. We can instantiate these using code, and can even write our own from scratch. But generally, we configure these using the storyboard.

The Main Run Loop

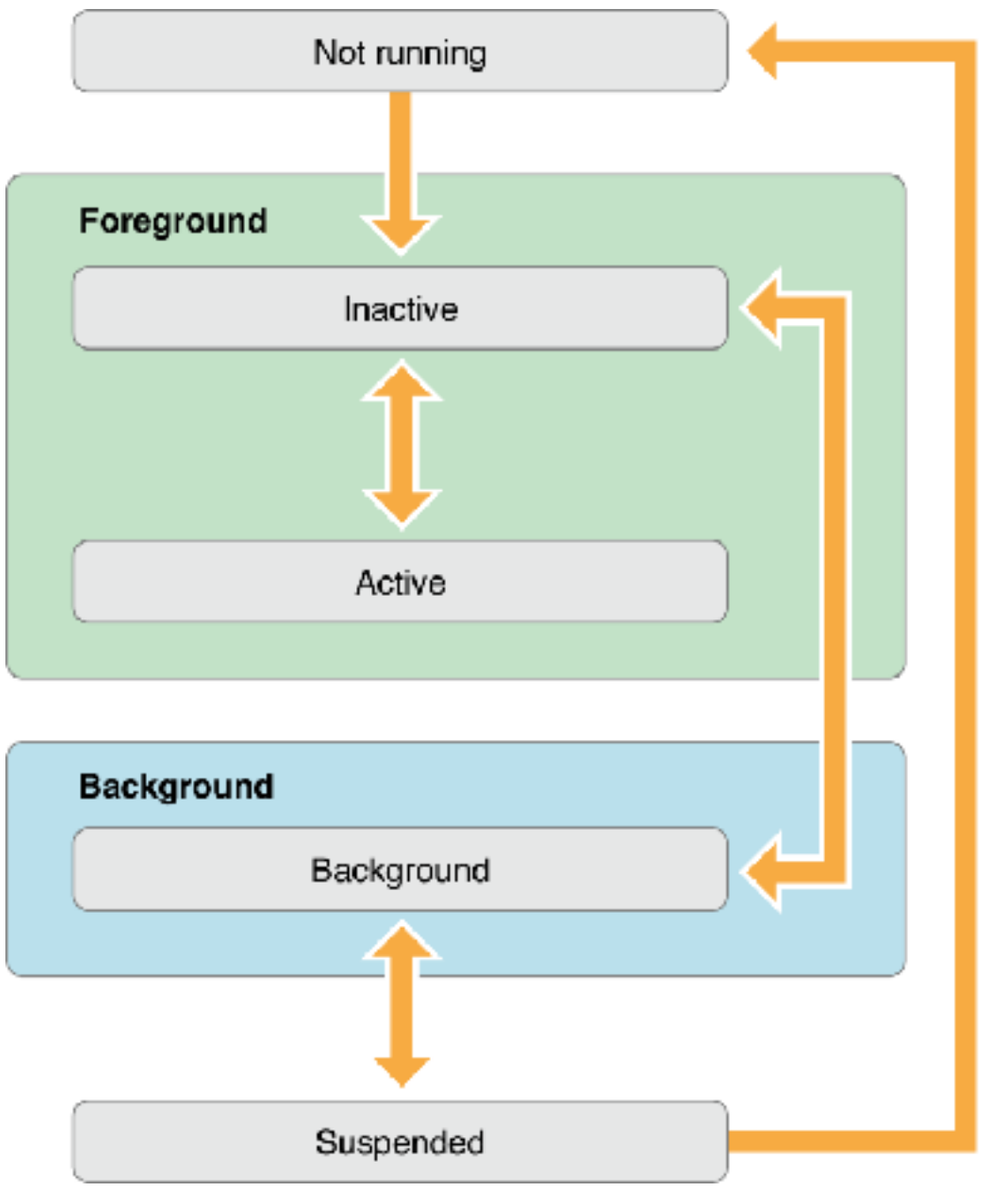


App States



Not Running	Not launched, or terminated
Inactive	In foreground, executing code (for example configuration at enter foreground) but not handling events
Active	In foreground and handling events
Background	In background, and executing code, but not user events. Normally a step towards suspended but can be a state the app enters to handle system events (location updates, playing media in background, updates from server etc)
Suspended	App in background and not executing code. Can exit to background to handle system events, and can be quickly restored. Small footprint in terms of resources.

AppDelegate methods to manage states



application:willFinishLaunchingWithOptions

applicationDidBecomeActive/applicationWillResignActive

applicationDidEnterBackground/applicationWillEnterForeground

applicationWillTerminate:

Next time…

- Storyboards and view controllers

Acknowledgements

- Several images used in these slides are taken from Apple’s iOS Development documentation and used as an educational aid.