Academy for iOS App Development

Dr. William C. Bulko

Core Location

Core Location

Core Location is a framework that provides the following services for a device to determine:

- its geographic location
- Its altitude
- Its orientation
- Its position relative to a nearby beacon.

The framework uses all available onboard hardware to do this.

- Wi-fi and/or cellular
- GPS
- Bluetooth
- Magnetometer
- Barometer

Core Location (cont.)

To begin using Core Location services, you must first import the framework, and then create a `CLLocationManager` object to start, stop, and manage the delivery of location-related events to your app.

```
import CoreLocation
     . . .
var locationManager = CLLocationManager()
```

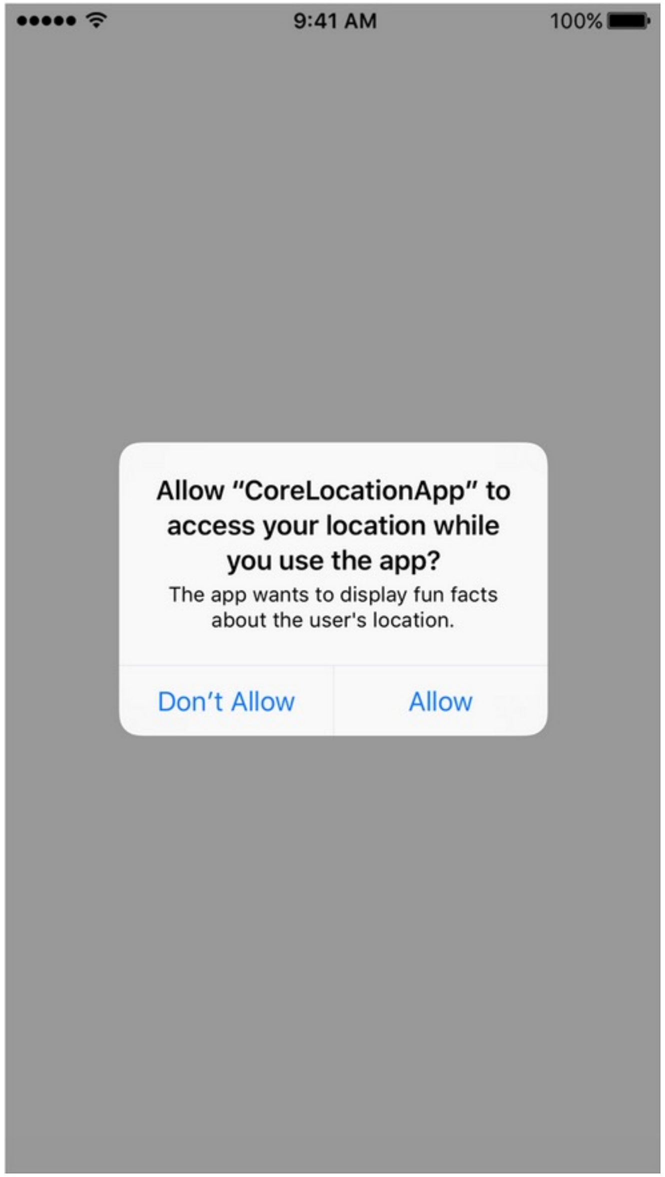
Then you must ask the user permission to use Location Services.

```
locationManager?.requestWhenInUseAuthorization()
```

Core Location (cont.)

This will cause the system to bring up an alert requesting access from the user.

Once the user responds, your app records the user’s response, and will not display this alert again.



9:41 AM 100%

Allow “CoreLocationApp” to access your location while you use the app?

The app wants to display fun facts about the user's location.

MapKit

`MapKit` is an API that makes it easy to display maps, plot locations, draw routes and other shapes on top of the map.

- Elements drawn on top of the map are called *overlays.*

You can choose what kind of map is rendered:

- Standard
- Satellite
- Hybrid

You can define pins (annotations) for locations of interest.

- You can define the color of pins.
- You can define *callouts* that show some basic identifying information when the pin is touched.

MapKit (cont.)

In order to display a map, you must:

- Import `MapKit`
- Include an `MKMapView` somewhere in a view controller

There are methods available for a number of things, including:

- Adding overlays
- Setting center of the map and the currently visible region of the map
- Changing the map type
- Using coordinates (latitude, longitude) for pin location and/or for a travel path
- Adding annotations

Academy for iOS App Development

Dr. William C. Bulko

Animation

Animation

What does it mean to *animate* something?

- To bring to life
- To cause to appear as if it's moving or changing

In the context of iOS applications, this means to modify aspects of the user interface in a special way as to produce the appearance of action.

Why would we want to animate something?

- It draws the user towards things that change
- It indicates importance at a particular moment
- It makes your app look cool, fun, or polished – which can be a differentiator

Animation

You can animate the following properties of a UIView derived object:

- **frame** – move or scale the view (relative to its superview)
- **bounds** – move the view’s contents within the view
- **center** – move the view relative to the screen
- **transform** - scale, rotate, or translate the view relative to its center point
- **alpha** - gradually change the transparency of the view
- **backgroundColor** - change the view’s background color
- **contentStretch** - change the way the view’s contents are stretched to fill the available space

UIView.animate

The basic UIView animation method is `UIView.animate`:

```
UIView.animate(
     withDuration: <duration>,
     delay: <delay>,
     options:  <options>,
     animations: {
          <animation code>
     }
     completion: {
          <completion code>
     }
)
```

`UIView.animate` (cont.)

`duration`: how long in seconds to run the animation

`delay`: how long to wait until starting the animation

`options`:

`.curveEaseInOut`	begin slow, accelerate, end slow
`.curveEaseIn`	begin slow, accelerate to end
`.curveEaseOut`	begin quickly, slow to end
`.curveLinear`	even over the duration
`.repeat`	make the animation loop forever
`.autoreverse`	animate forward, then reverse

`animation code`:

identifies the ending value for the selected attribute(s)

`completion code`:

code to be executed at the end of the animation

Alpha Animation: Fade Out

Adjust the *alpha*:

```
// Starting alpha value
self.labelName.alpha = 1.0

UIView.animate(
      withDuration: 3.0,
      animations: {
           self.labelName.alpha = 0.0
      }
)
```

Center Animation: Slide Out to Right

Adjust the *center*:

```
// Starting center value
self.labelName.center.x = self.view.center.x

UIView.animate(
     withDuration: 3.0,
     animations: {
          self.labelName.center.x +=
                self.view.bounds.width
     }
)
```

Center Animation: Spinning

Adjust the *transform*:

```
UIView.animate(
    withDuration: 3.0,
    animations: {
        // 180 degree rotation
        self.labelName.transform =
            self.labelName.transform.rotated(
                 by: CGFloat(Double.pi))
        }
)
```