



Core Location

Giftbot

Core Location Framework

기기의 지리적 위치와 방향을 얻기 위한 프레임워크

- 기기의 지리적 위치, 고도, 방향 또는 iBeacon 주변의 상대적 위치를 결정하는 서비스를 제공
특정 위치에 도달하면 작동
 - 다음과 같은 모든 사용 가능한 온보드 하드웨어를 이용해 데이터를 수집
Wi-Fi, GPS, Bluetooth, Magnetometer (자력계), Barometer (기압계), Cellular Hardware 등
 - CLLocationManager 를 이용해 대부분의 서비스를 시작하고 연결된 Delegate 를 통해 응답을 수신
- ★• 위치 정보를 얻기 위해 반드시 유저로부터 권한을 얻어야 함
- 크게 위치 업데이트 / 지역 모니터링 / iBeacon / 장치 방향 / 좌표 변환 등의 역할 수행

Request Authorization

Requesting Authorization for Location Services

`func requestWhenInUseAuthorization()`

Requests permission to use location services while the app is in the foreground.

`func requestAlwaysAuthorization()`

Requests permission to use location services whenever the app is running.

`enum CLAuthorizationStatus`

Constants indicating whether the app is authorized to use location services.

Request Authorization

When-in-use authorization

- : 앱이 **Foreground**에서 동작 중일 때만 위치 서비스 사용
- : 앱을 자동으로 재실행하는 서비스는 사용 불가
- : 반드시 **Always**를 써야 하는 경우가 아니면 이 방식을 권장. 동시에 한 가지 방식으로만 설정 가능

Always authorization

- : **Foreground** 나 **Background** 모두에서 필요할 때 위치 서비스 사용
- : 앱이 실행 중이지 않을 때 위치 기반 이벤트가 발생하면 시스템이 앱을 실행하고 이벤트 전달

```
let locationManager = CLLocationManager()  
locationManager.requestWhenInUseAuthorization()  
locationManager.requestAlwaysAuthorization()
```

Info.plist - Usage Description

Always authorization 이용 시

iOS 11 이상 - NSLocationAlwaysAndWhenInUseUsageDescription 키 등록

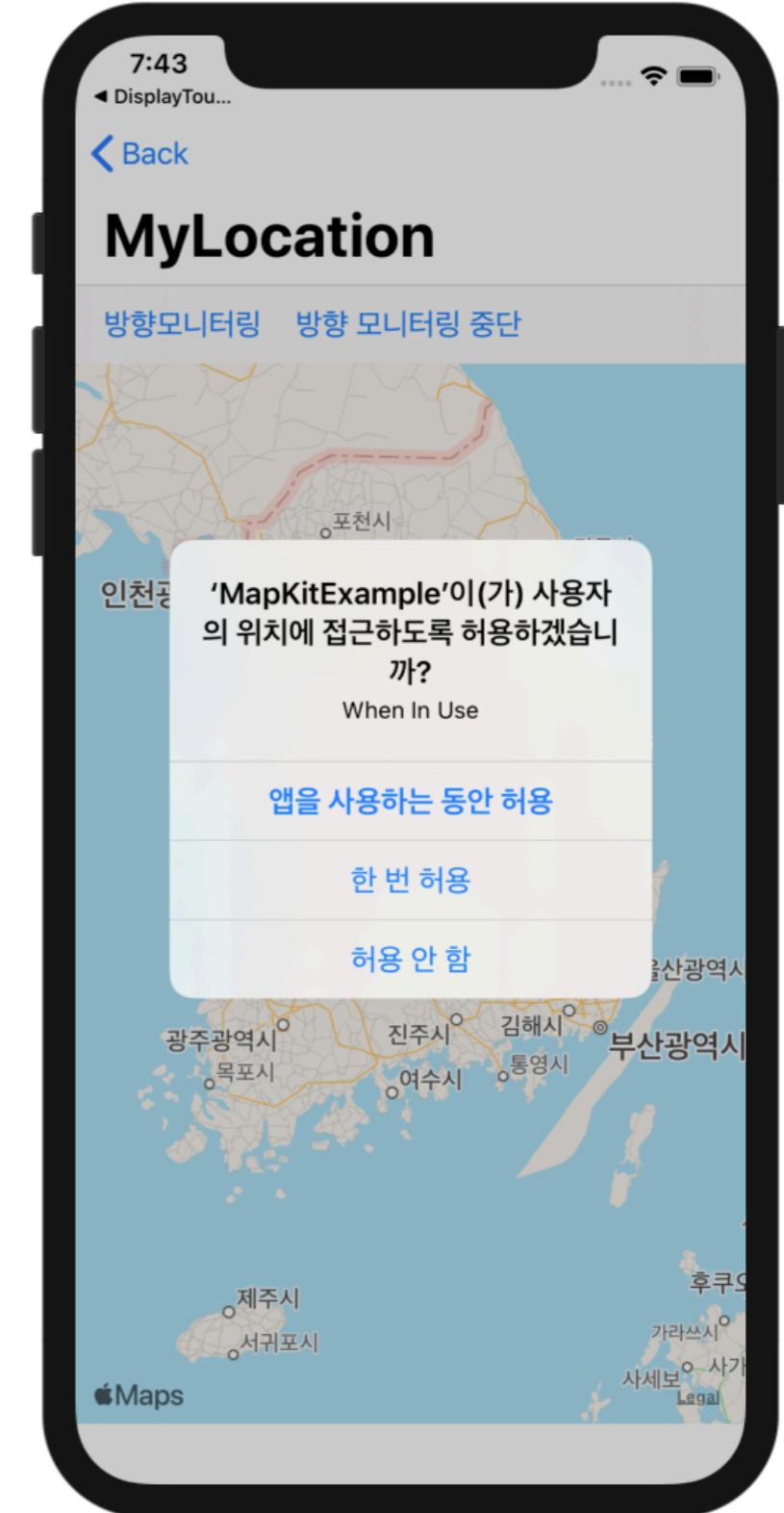
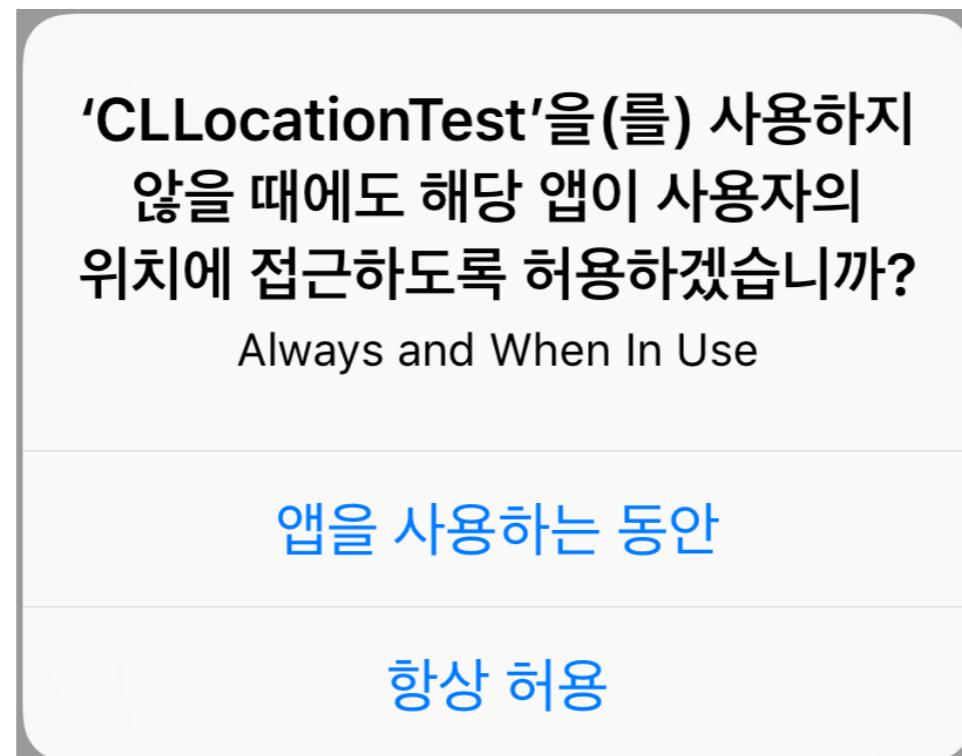
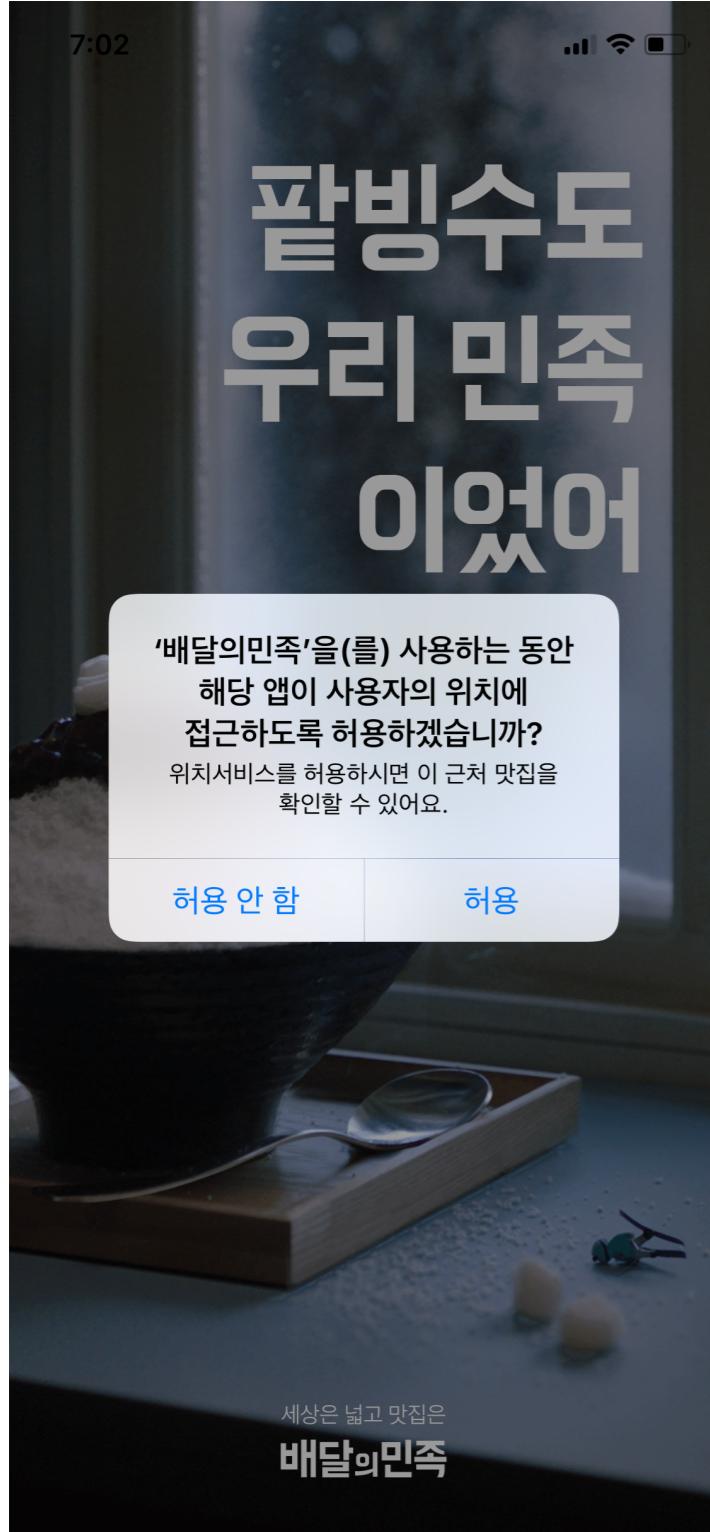
iOS 10 이하 - NSLocationAlwaysUsageDescription 키 등록

When-in-use authorization - NSLocationWhenInUseUsageDescription 키 등록

Key	Type	Value
▼ Information Property List	Dictionary	(17 items)
Privacy - Location Always and When In Use Usage Description	String	Always and When In Use
Privacy - Location Always Usage Description	String	Always
Privacy - Location When In Use Usage Description	String	When In Use

3가지를 다 넣어야 10버전 이하까지 모두 사용 가능함.

Request Authorization



Allow Once

Option	Authorization
Allow While Using App	When In Use authorization that does not expire.
Allow Once	Temporary When In Use authorization that expires when the app is no longer in use.
Don't Allow	Denied; no further authorization requests are allowed.

Service	When-in-use	Always
Standard location service	Supported	Supported
Significant-change location service <small>사용자의 위치가 대폭 변화했을 때만 업데이트 제공</small>	Not available	Supported
Visits service <small>한 장소에 머물다가 이동할 때 업데이트 제공</small>	Not available	Supported
Region monitoring	Not available	Supported
iBeacon ranging	Supported	Supported
Heading service	Supported	Supported
Geocoding services	Supported	Supported

Service	Launches app
	App을 깨울 수 있는지 여부

Standard location service	No
---------------------------	----

Significant-change location service	Yes
-------------------------------------	-----

Visits service	Yes
----------------	-----

Region monitoring	Yes
-------------------	-----

iBeacon ranging	No
-----------------	----

Heading service	No
-----------------	----

CLAuthorizationStatus

```
switch CLLocationManager.authorizationStatus() {  
    case .notDetermined: 사용자가 권한 허용을 한 번도 안한 경우 → 권한 허용 alert 사용  
        locationManager.requestWhenInUseAuthorization()  
    case .restricted, .denied:  
        // Disable location features  
        disableMyLocationBasedFeatures()  
    case .authorizedWhenInUse:  
        // Enable basic location features  
        enableMyWhenInUseFeatures()  
    case .authorizedAlways:  
        // Enable any of your app's location features  
        enableMyAlwaysFeatures()  
}
```

Delegate - didChangeAuthorization

상태 값이 변경되면 호출되는 함수

```
func locationManager(  
    _ manager: CLLocationManager,  
    didChangeAuthorization status: CLAuthorizationStatus  
) {  
  
    switch status {  
    case .authorizedWhenInUse, .authorizedAlways:  
        print("Authorized")  
    default:  
        print("Unauthorized")  
    }  
}
```

Determining the Availability

위치 서비스를 사용할 수 없는 상황

- 기기에 이 기능을 지원하는 데 필요한 하드웨어가 없음
- 사용자가 시스템 설정에서 위치 서비스 기능을 끔
- 사용자가 앱의 위치 서비스에 대한 접근을 거부
- 기기가 비행기 모드로 설정되었을 때
- 백그라운드 갱신 기능을 사용할 수 없고 필요한 기능을 우선 순위가 높은 다른 서비스가 이미 사용 중일 때

위치 서비스를 사용하지 못하는 경우가 있을 수 있기 때문에 아래 코드들을 활용하여 가용성 체크를 한 후 앱을 실행하도록 해야 함.

가용성 체크

```
func locationServicesEnabled() -> Bool
func headingAvailable() -> Bool
func significantLocationChangeMonitoringAvailable() -> Bool
func isMonitoringAvailable(for regionClass: Swift.AnyClass) -> Bool
func isRangingAvailable() -> Bool
```

Getting the User's Location Data

Standard location service (When-in-use or Always authorization)

- 사용자 위치를 실시간으로 파악하기 위한 범용 솔루션
- 다른 서비스에 비해 더 많은 전력을 쓰지만 가장 정확하고 즉각적인 정보를 제공

Significant-change location service (Always authorization)

- 전력 소모를 줄이기 위한 것으로 업데이트가 자주 필요하지 않고 GPS 정밀도가 낮아도 되는 경우 사용
- 사용자 위치를 대폭 변경한 경우에만 업데이트를 제공
- 사용자가 걷고 있을 때 주변의 관심 장소(POI)에 대한 추천 정보를 제안해주는 등의 서비스를 제공 가능

Visits location service (Always authorization)

- 가장 효율적으로 전력을 사용하지만 다른 서비스에 비해 업데이트 횟수가 적은 방법
- 유저가 한 장소에 머물러 시간을 보내다가 이동할 때 업데이트 알림 발생 (위치 및 시간 정보)
- 사용자의 행동 패턴을 파악하고 그 지식을 앱의 다른 부분에 적용하기 위한 서비스로 활용

Standard Location Service

지정한 설정 값에 따라 정기적인 위치 업데이트를 수행

다른 위치 서비스와 달리 When-In-Use 권한에서도 사용 가능

위치 정보를 세밀하게 파악해야 하고 자주 업데이트 해야 하는 앱에서 사용

높은 정밀도를 필요로 하지 않을 경우 시스템이 알아서 GPS 대신 Wi-Fi 를 이용하는 등의 작업 수행

```
func startUpdatingLocation() {  
    let status = CLLocationManager.authorizationStatus()  
    guard status == .authorizedAlways || status == .authorizedWhenInUse,  
        CLLocationManager.locationServicesEnabled() // location 서비스가 가능한지 확인  
    else { return }  
    // 사용 조건 설정  
    locationManager.desiredAccuracy = kCLLocationAccuracyNearestTenMeters  
    locationManager.distanceFilter = 10.0  
    locationManager.startUpdatingLocation() // 업데이트 실행되고 delegate로 정보가 들어오기 시작  
}
```

desiredAccuracy

위치 정보의 정밀도 값으로, 반드시 보장되지는 않지만 가능한 한 설정에 가까운 정보를 취합

정밀도가 높을수록 더 많은 전력 소비와 연산 시간이 필요하므로 높은 것이 꼭 좋은 것은 아님

높은 정밀도의 위치 정보 요청 시, 우선 가능한 빨리 초기 정보를 받은 뒤 더 정확한 위치 정보를 계속 파악해

더 정확한 정보를 갱신하는 형태로 전달받게 됨

기본값 - AccuracyBest (iOS, macOS) / AccuracyHundredMeters (watchOS)

```
locationManager.desiredAccuracy = kCLLocationAccuracyBest  
                               기본값
```

```
typealias CLLocationAccuracy = Double  
let kCLLocationAccuracyBestForNavigation: CLLocationAccuracy  
let kCLLocationAccuracyBest: CLLocationAccuracy  
let kCLLocationAccuracyNearestTenMeters: CLLocationAccuracy  
let kCLLocationAccuracyHundredMeters: CLLocationAccuracy  
let kCLLocationAccuracyKilometer: CLLocationAccuracy  
let kCLLocationAccuracyThreeKilometers: CLLocationAccuracy
```

distanceFilter

위치 업데이트 이벤트를 발생시키기 위해 필요한 기기의 최소 이동 거리 (Meter 단위) 설정

기본값 - `kCLDistanceFilterNone` (모든 이동 감지시마다 업데이트)

Standard Location Service 를 사용할 때는 `desiredAccuracy` 와 `distanceFilter` 를 설정 필요
desired accuracy값과 서로 영향을 주고 받음. 둘 다 정밀하거나 둘 다 아니거나 해야 함.

```
locationManager.distanceFilter = kCLDistanceFilterNone
```

```
typealias CLLocationDistance = Double
```

```
let kCLDistanceFilterNone: CLLocationDistance
```

```
let CLLocationDistanceMax: CLLocationDistance
```

Represent Location Data

class **CLLocation**

The latitude, longitude, and course information reported by the system.

struct **CLLocationCoordinate2D** 위 경도값. (좌표값)

The latitude and longitude associated with a location, specified using the WGS 84 reference frame.

class **CLFloor** 층별로 업데이트 제공. 높은 빌딩의 층별 정보 제공 하는 경우.

The floor of a building on which the user's device is located.

class **CLVisit**

Information about the user's location during a specific period of time.

CLLocation

시스템으로부터 전달되는 위도, 경도, 고도 및 코스 정보 등을 담고 있는 객체

그대로 사용하도록 설계되었으므로 서브클래싱 하지 않아야 하며, 일반적으로 직접 생성하지 않음

커스텀 위치 정보를 캐시하고 싶거나 두 지점 사이의 거리를 구하는 경우 등에는 직접 생성할 수도 있음

```
func locationManager(  
    _ manager: CLLocationManager,  
    didUpdateLocations locations: [CLLocation]) {  
    let lastLocation = locations.last!  
    let location = CLLocation(latitude: lastLocation.latitude,  
                               longitude: lastLocation.longitude)  
    let distance = lastLocation.distance(from: location)  
    print(distance)  
}
```

배열형태로 엘리먼트 간 연산을 통해 거리 구할 수 있음.

특정 지점과의 거리

CLLocation

[위치]

```
var coordinate: CLLocationCoordinate2D { get }
var altitude: CLLocationDistance { get }
var horizontalAccuracy: CLLocationAccuracy { get }
var verticalAccuracy: CLLocationAccuracy { get }
var floor: CLFloor? { get }
var timestamp: Date { get }
```

[좌표 간 거리]

```
func distance(from location: CLLocation) -> CLLocationDistance
```

[속도 및 코스 정보]

```
var course: CLLocationDirection { get }
var speed: CLLocationSpeed { get }
```

CLLocationCoordinate2D

WGS 84 참조 프레임을 사용하여 지정된 위도, 경도에 대한 위치 정보

※ WGS : World Geodetic System. 1984년에 제정된 범 지구적 측위 시스템

[좌표]

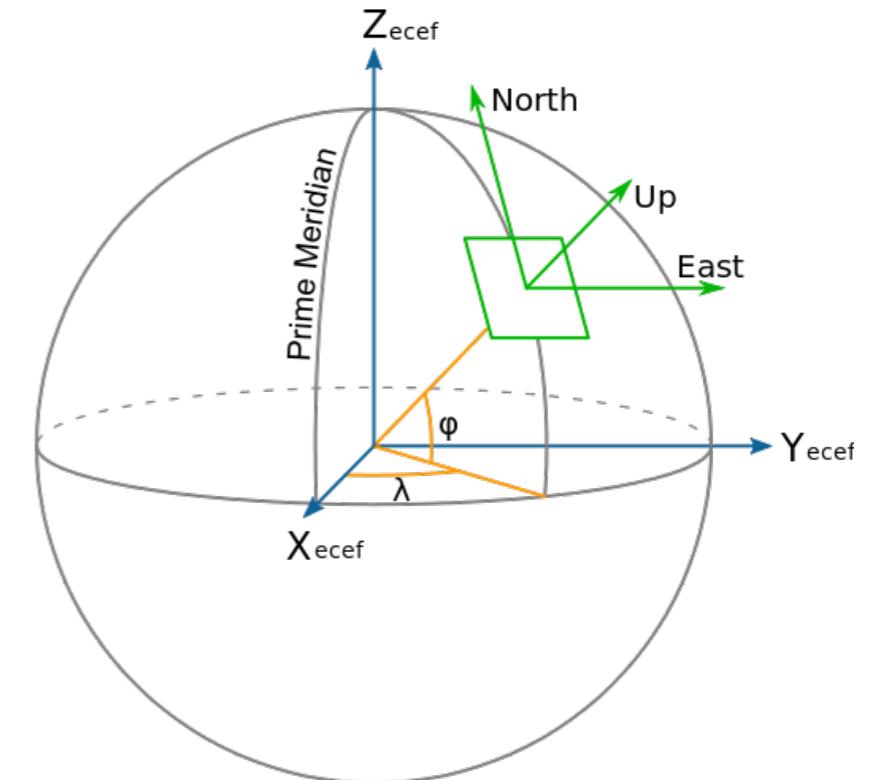
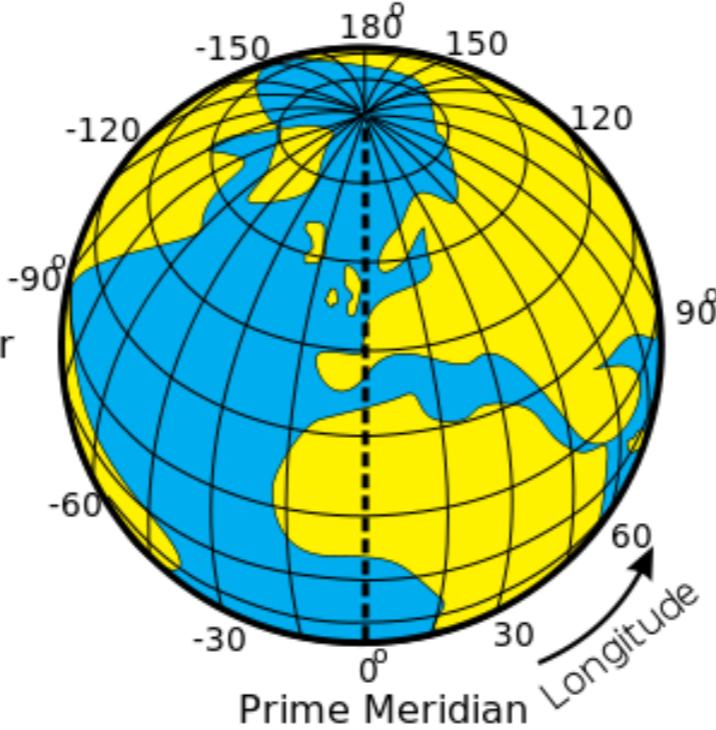
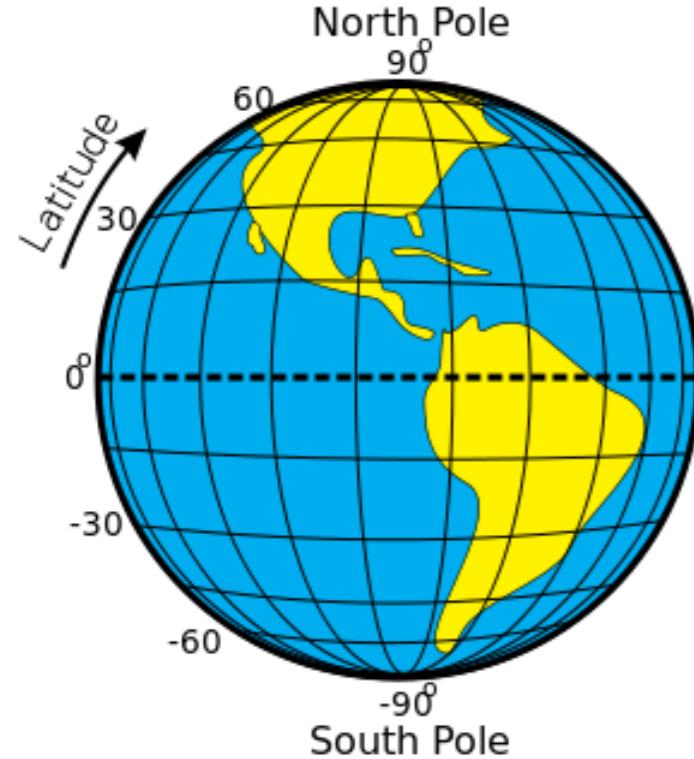
```
var latitude: CLLocationDegrees  
var longitude: CLLocationDegrees
```

[관련 함수] - CLLocation.h 에 정의

```
func CLLocationCoordinate2DMake(  
    _ latitude: CLLocationDegrees, _ longitude: CLLocationDegrees  
) -> CLLocationCoordinate2D
```

```
func CLLocationCoordinate2DIsValid(  
    _ coord: CLLocationCoordinate2D  
) -> Bool
```

Geographic coordinate system



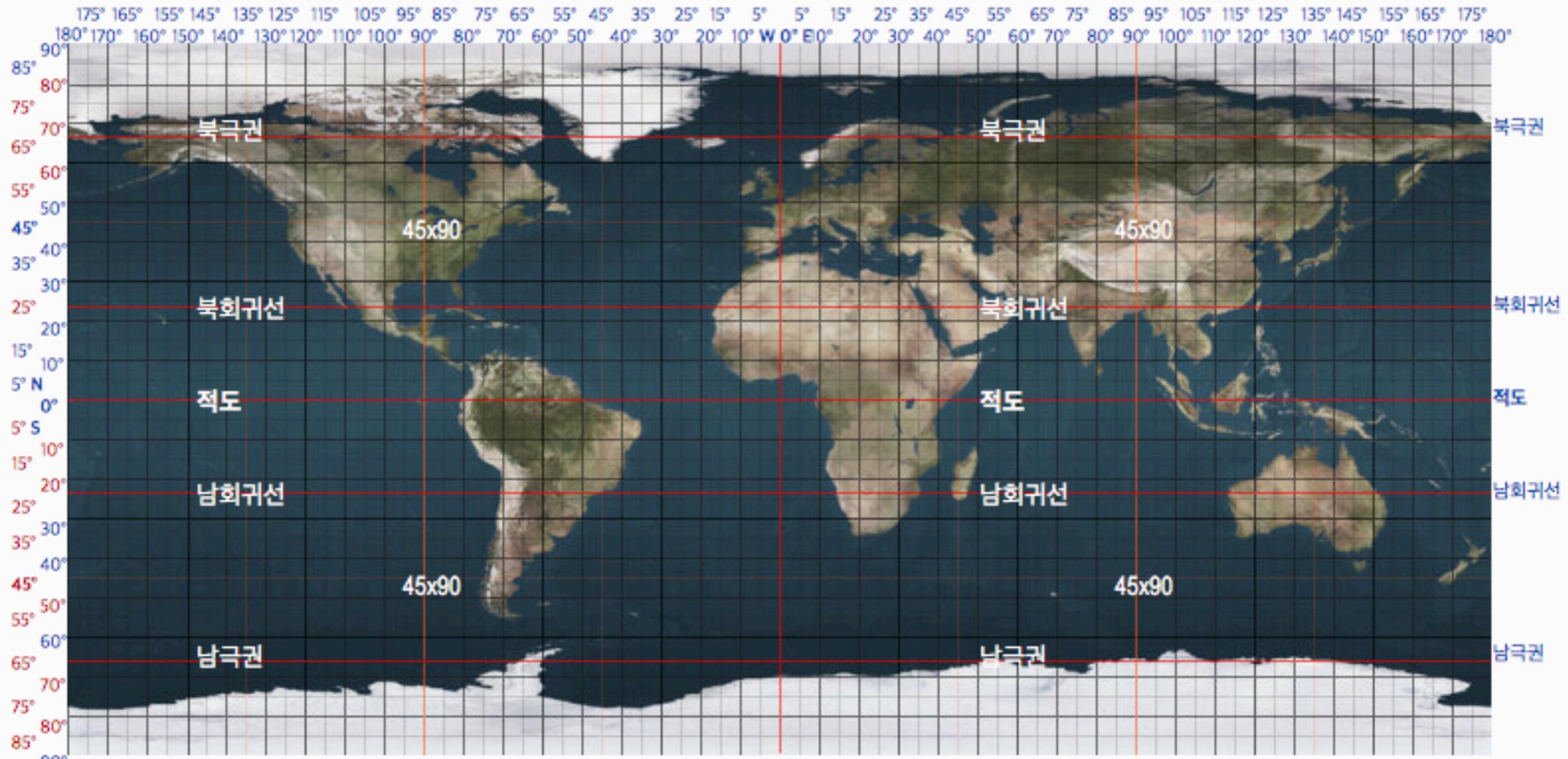
latitude [lætɪtu:d] : 위도

- 적도면과 표면 지점의 수직선(추선)이 이루는 각
- 범위 90 ~ -90 / 북위 90 (북반구) ~ 남위 90 (남반구)

longitude [la:ndʒətu:d] : 경도

- 북극부터 남극까지의 표면 지점을 그은 경선과 본초 자오선 (현재 그리니치 천문대 기준)이 이루는 각
- 범위 180 ~ -180, 동경 180 (동반구) ~ 서경 180 (서반구)

Geographic coordinate system



Geographic coordinate system

서울 시청 - 37° 33' 58.87"N / 126° 58' 40.63"E

* 읽는 법

- 37도 33분 58.87초
- thirty-seven degrees thity-three minutes fifty-eight dot eighty-seven seconds north

* 계산 방법 ($1^\circ = 60'$, $1' = 60''$)

$$37.5663528 = 37^\circ 33' 58.87" N$$

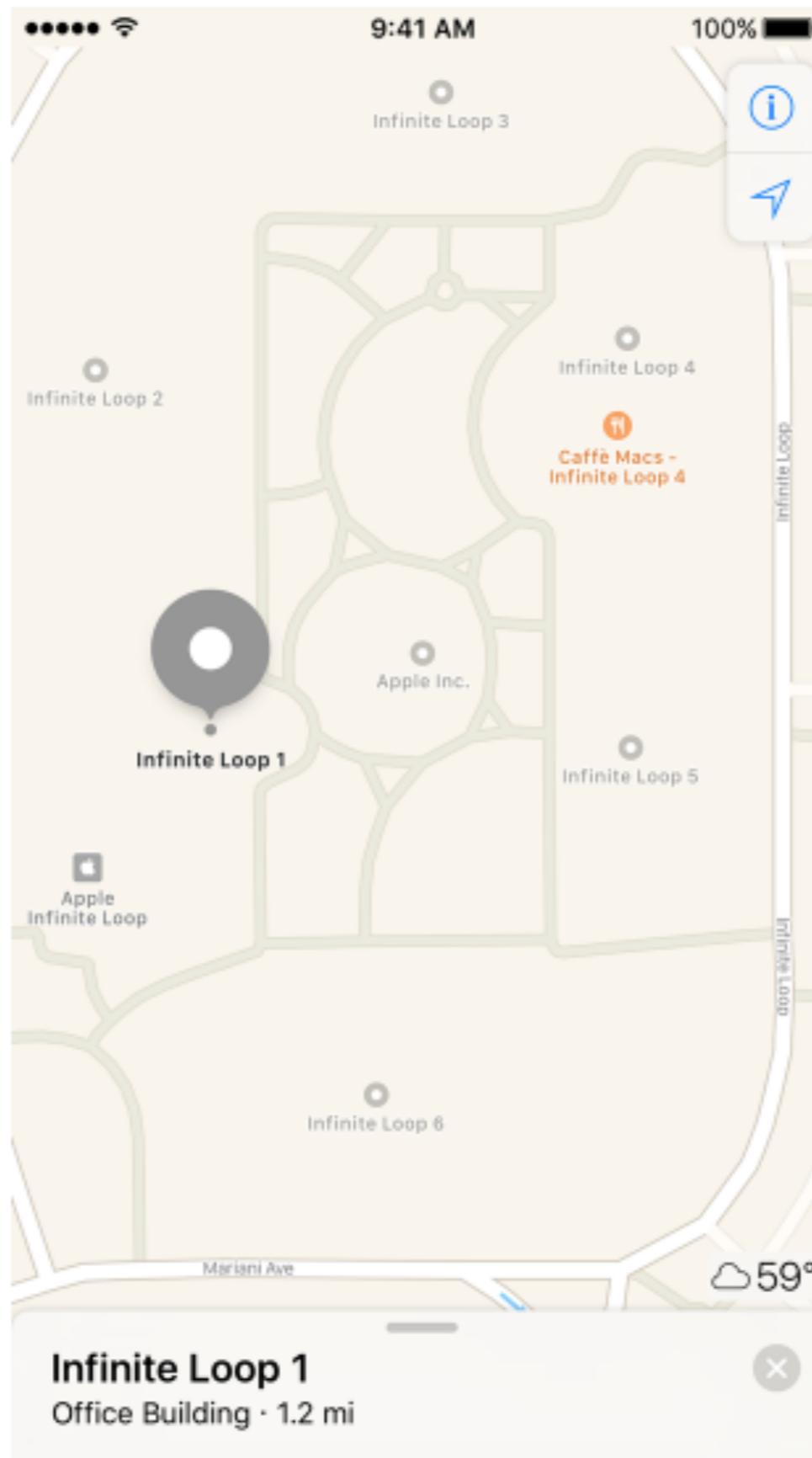
e.g.

$$37 = 37^\circ$$

$$0.5663528 * 60 = 33.981168'$$

$$0.981168 * 60 = 58.87"$$

Geocoding



Geocoding

CLGeocoder 를 이용해 지리적 좌표(위도/경도)와 사용자에게 친숙한 지명 간 변환 처리
장소에 대한 정보는 CLPlacemark 를 통해 제공

Geocoder 객체는 단일 객체(one-shot objects) 로서, 각 객체를 하나의 변환에만 사용
여러 오브젝트를 생성해 다중 변환을 수행할 수 있으나 그 수에 제한이 있으며 그 이상 시도 시 실패 가능성

Reverse Geocode : Coordinate → Placemark (좌표를 지명으로)

Geocode : Placemark → Coordinate (지명을 좌표로)

[Geocoding 을 효과적으로 사용하기 위한 팁]

한 사용자 액션에 대해 하나의 Geocoding 만 요청

동일 또는 근접 거리에서는 새로운 지오 코딩을 수행하는 대신 기존 요청 결과를 재활용

일반적인 상황에서는 분당 하나 이상의 지오 코딩 요청을 보내지 말 것

앱이 Foreground 상태여서 사용자가 결과를 즉시 볼 수 있는 상황에서만 Geocoding 수행

Reverse Geocode Location

좌표를 지명으로 변환

```
func lookUpCurrentLocation(  
    completionHandler: @escaping (CLPlacemark?) -> Void  
) {  
    guard let lastLocation = locationManager.location else {  
        return completionHandler(nil)  
    }  
    let geocoder = CLGeocoder()  
    geocoder.reverseGeocodeLocation(lastLocation) {  
        (placemarks, error) in  
        let firstLocation = placemarks?.first  
        completionHandler(firstLocation)  
    }  
}
```

Geocode Location

지명을 좌표값으로 변환

```
func getCoordinate(  
    addressString: String,  
    completionHandler: @escaping(CLLocationCoordinate2D, Error?) -> Void  
) {  
    let geocoder = CLGeocoder()  
    geocoder.geocodeAddressString(addressString) {  
        (placemarks, error) in  
        guard error == nil else {  
            return completionHandler(kCLLocationCoordinate2DInvalid, error)  
        }  
        if let placemark = placemarks?.first {  
            completionHandler(placemark.location!.coordinate, nil)  
        }  
    }  
}
```