

# What's New in Core Location

Session 307

**Jay Bruins**

Software Engineer

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# What We Will Cover

# What We Will Cover

- Multitasking impact on location

# What We Will Cover

- Multitasking impact on location
- Creating fitness applications

# What We Will Cover

- Multitasking impact on location
- Creating fitness applications
- New region monitoring features

# This Is Not a Review

## Resources for getting started

- Staying on Track with Location Services (WWDC 2012)
- Location Awareness Programming Guide

# Multitasking Impact on Location

# Multitasking Impact on Location

User empowerment



# Multitasking Impact on Location

## User empowerment

- Disable multitasking

# Multitasking Impact on Location

## User empowerment

- Disable multitasking
- Disable multitasking for your app

# Multitasking Impact on Location

## User empowerment

- Disable multitasking
- Disable multitasking for your app
- Quit app from app switcher

# Multitasking Impact on Location

Core Location APIs

# Multitasking Impact on Location

## Core Location APIs

- Continuous background location

# Multitasking Impact on Location

## Core Location APIs

- Continuous background location
- Significant location change

# Multitasking Impact on Location

## Core Location APIs

- Continuous background location
- Significant location change
- Region monitoring

# Multitasking Impact on Location

Recommendations



# Multitasking Impact on Location

## Recommendations

- Be clear about why

# Multitasking Impact on Location

## Recommendations

- Be clear about why
- Use location selectively

# Multitasking Impact on Location

## Recommendations

- Be clear about why
- Use location selectively
- Don't start on every launch

# Multitasking Impact on Location

## Recommendations

- Be clear about why
- Use location selectively
- Don't start on every launch
- Minimize usage in background

# Multitasking Impact on Location

## Recommendations

- Be clear about why
- Use location selectively
- Don't start on every launch
- Minimize usage in background
  - Especially when fetching

# Multitasking Impact on Location

## Recommendations

- Be clear about why
- Use location selectively
- Don't start on every launch
- Minimize usage in background
  - Especially when fetching
- Turn off as appropriate

# Multitasking Impact on Location

## Recommendations

- Be clear about why
- Use location selectively
- Don't start on every launch
- Minimize usage in background
  - Especially when fetching
- Turn off as appropriate
  - Use a timer

# Multitasking Impact on Location

Limited continuous location



# Multitasking Impact on Location

Limited continuous location

- `UIBackgroundModes` key with `location` value

# Multitasking Impact on Location

Limited continuous location

- `UIBackgroundModes` key with `location` value
- Must start in foreground

# Multitasking Impact on Location

## Limited continuous location

- `UIBackgroundModes` key with `location` value
- Must start in foreground
- Or use significant location change

# Multitasking Impact on Location

Conflicting intentions

# Multitasking Impact on Location

## Conflicting intentions

- @property UIBackgroundRefreshStatus backgroundRefreshStatus

# Multitasking Impact on Location

## Conflicting intentions

- @property UIBackgroundRefreshStatus backgroundRefreshStatus
  - UIBackgroundRefreshStatusAvailable

# Multitasking Impact on Location

## Conflicting intentions

- @property UIBackgroundRefreshStatus backgroundRefreshStatus
  - UIBackgroundRefreshStatusAvailable
  - UIBackgroundRefreshStatusDenied

# Multitasking Impact on Location

## Conflicting intentions

- @property UIBackgroundRefreshStatus backgroundRefreshStatus
  - UIBackgroundRefreshStatusAvailable
  - UIBackgroundRefreshStatusDenied
  - UIBackgroundRefreshStatusRestricted



# Multitasking Impact on Location

## Conflicting intentions

- @property UIBackgroundRefreshStatus backgroundRefreshStatus
  - UIBackgroundRefreshStatusAvailable
  - UIBackgroundRefreshStatusDenied
  - UIBackgroundRefreshStatusRestricted
- UIApplicationBackgroundRefreshStatusDidChange

# Creating Fitness Applications

# Creating Fitness Applications

# Creating Fitness Applications

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];
```

# Creating Fitness Applications

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];  
locationManager.delegate = self;
```

# Creating Fitness Applications

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];  
locationManager.delegate = self;  
locationManager.activityType = CLActivityTypeFitness;
```

# Creating Fitness Applications

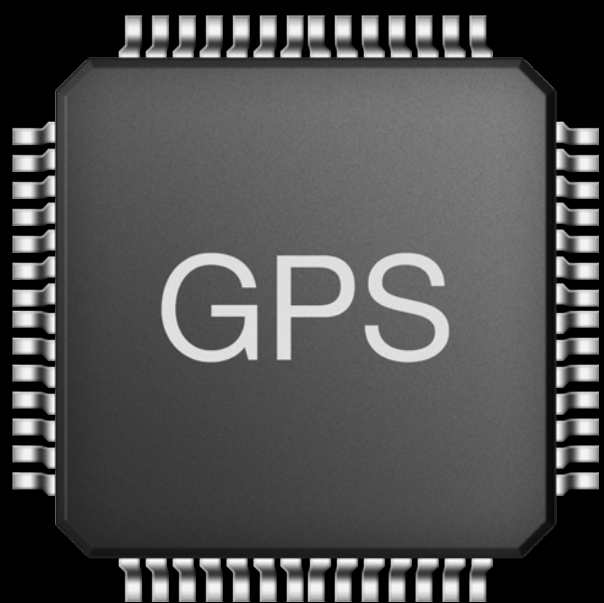
```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];  
locationManager.delegate = self;  
locationManager.activityType = CLActivityTypeFitness;  
[locationManager startUpdatingLocations];
```

# How GPS Works on iOS

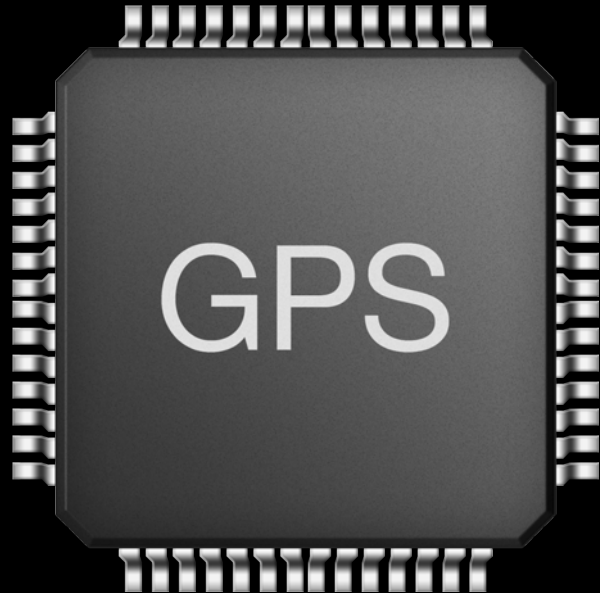
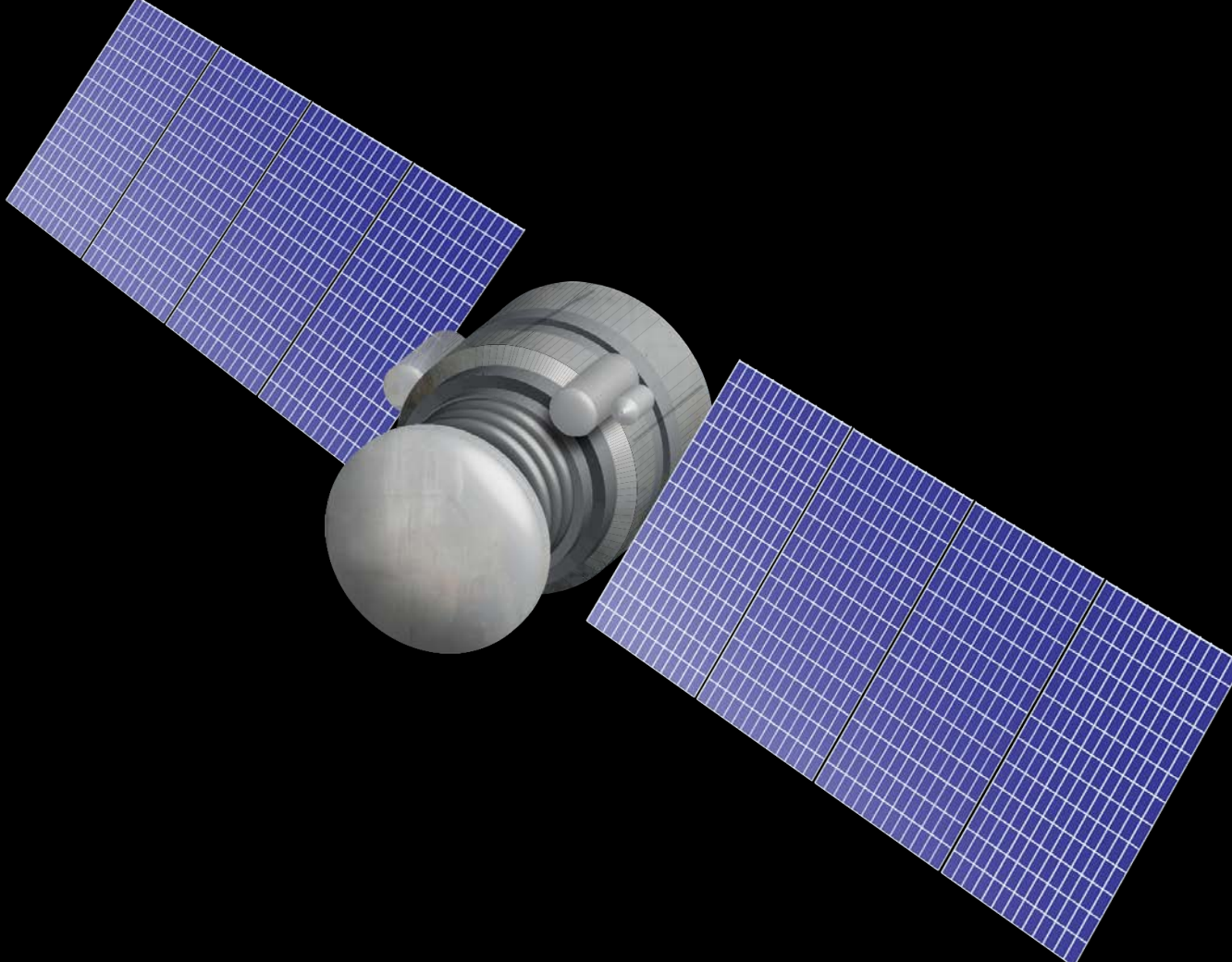




# How GPS Works on iOS

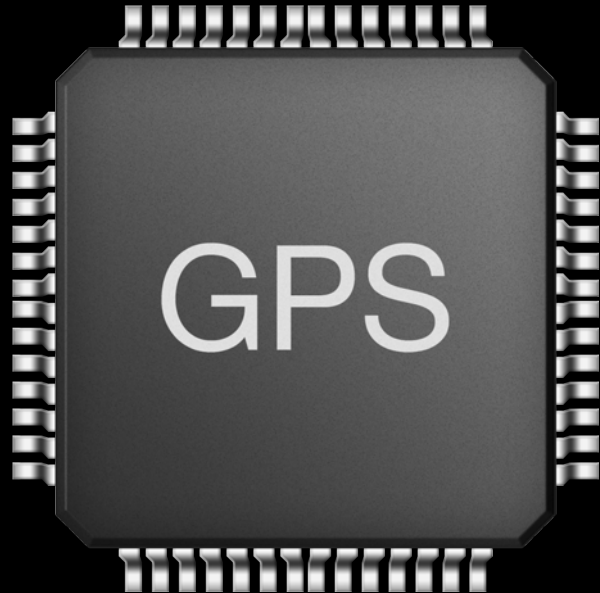
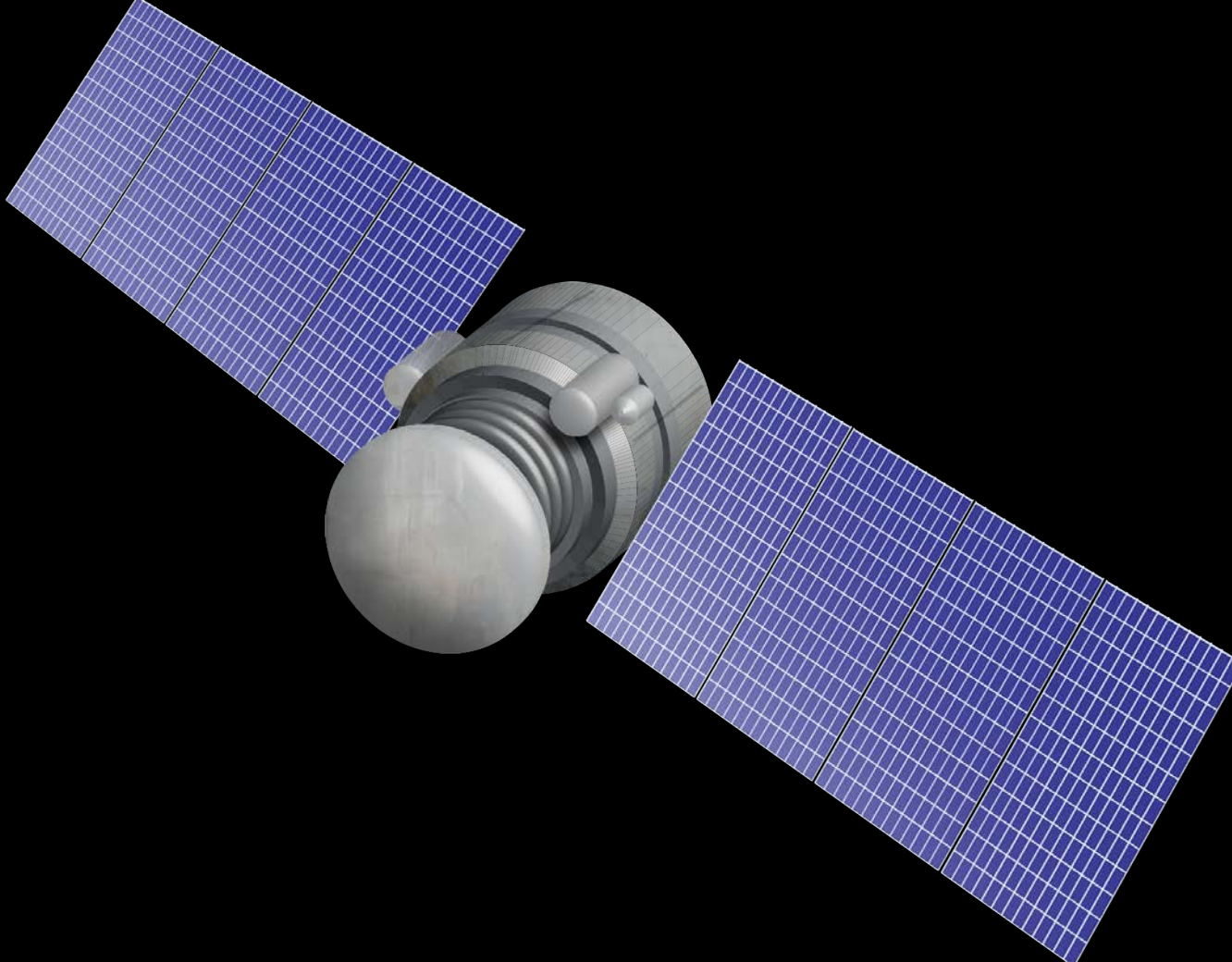


# How GPS Works on iOS

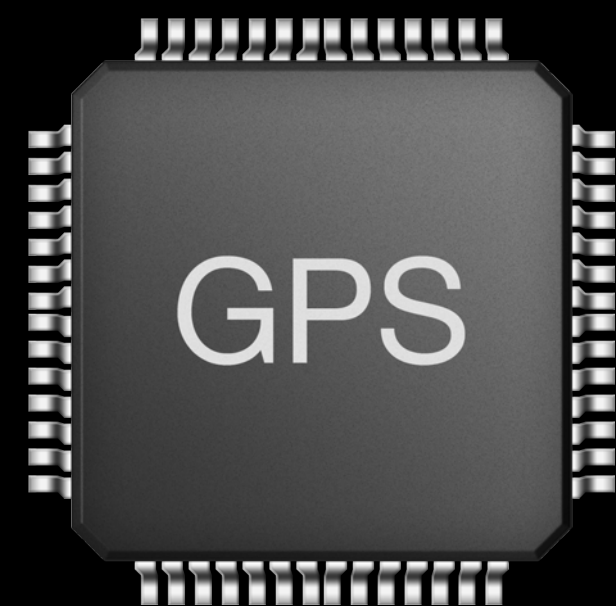
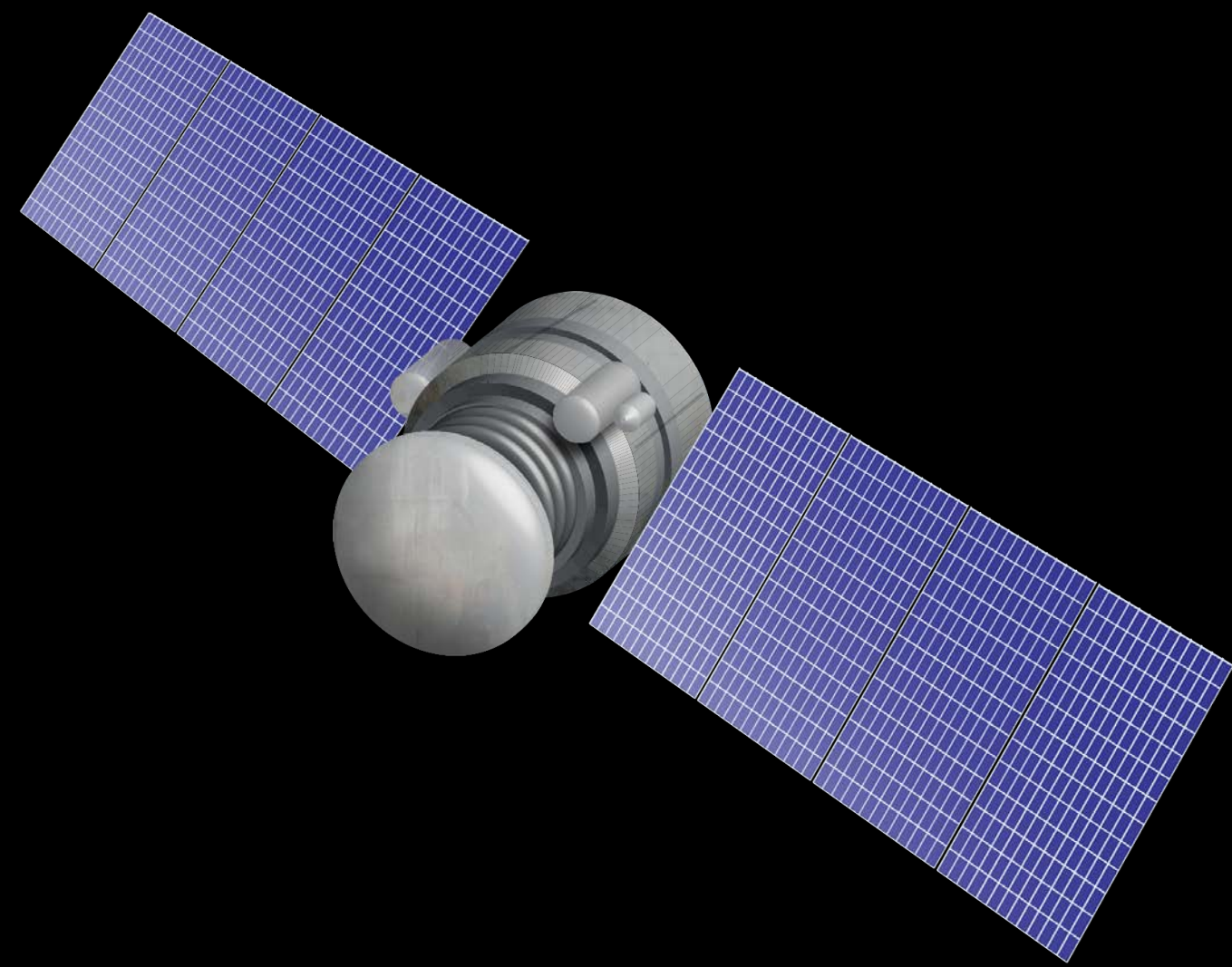




# How GPS Works on iOS

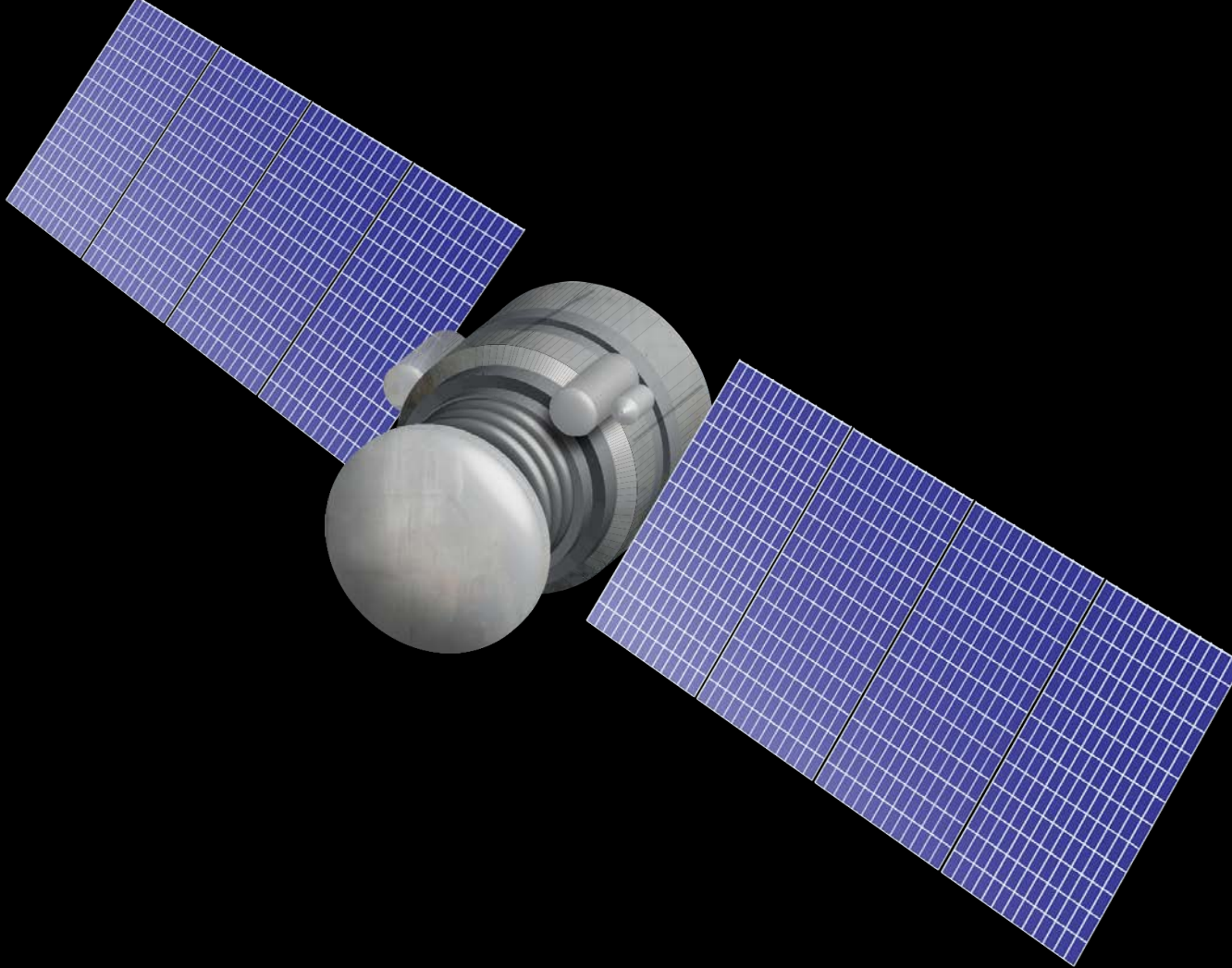


# How GPS Works on iOS





# How GPS Works on iOS

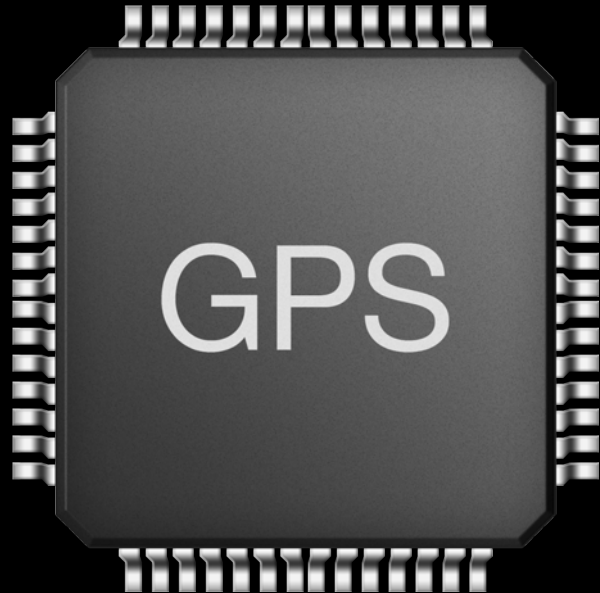
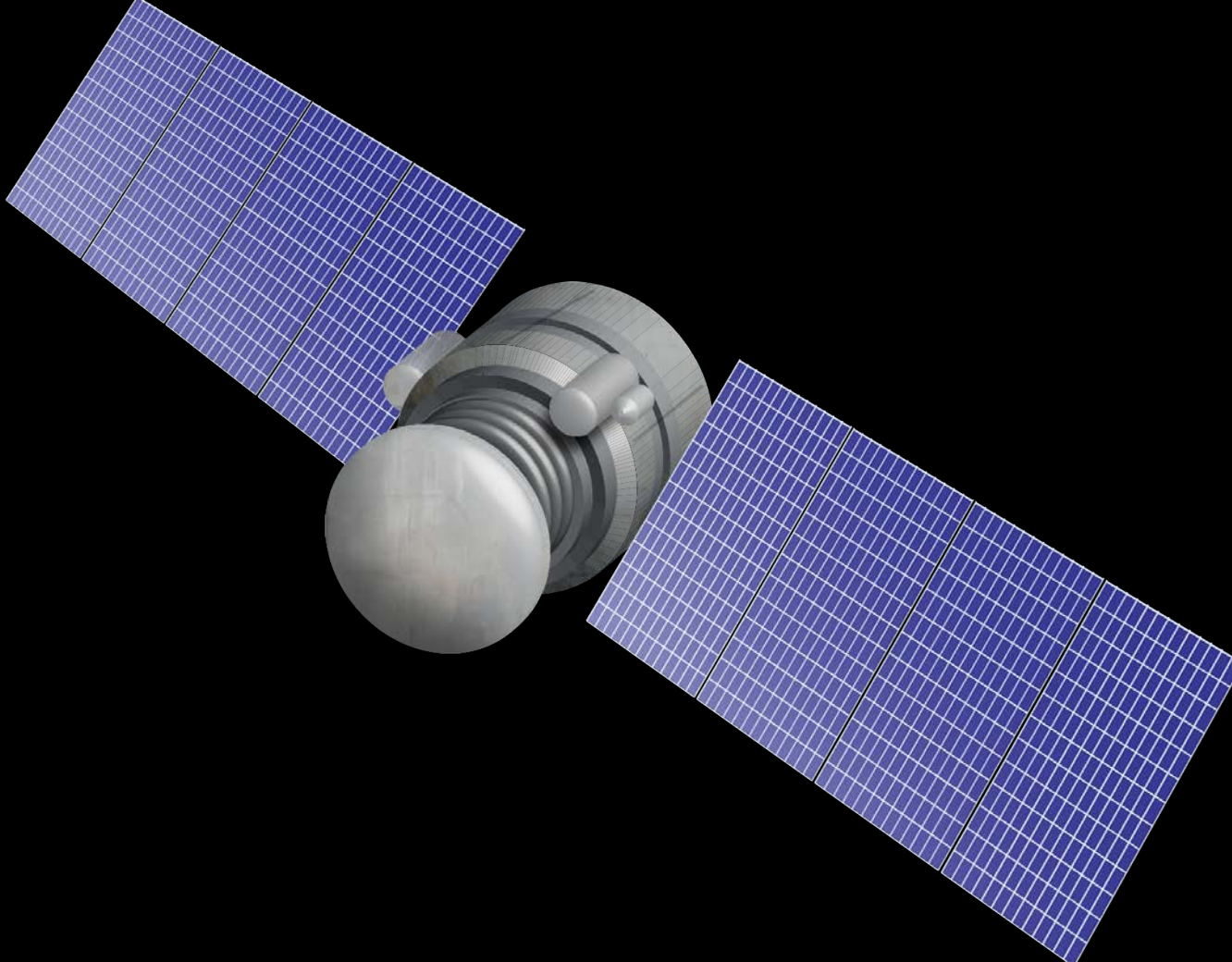


# How GPS Works on iOS





# How GPS Works on iOS

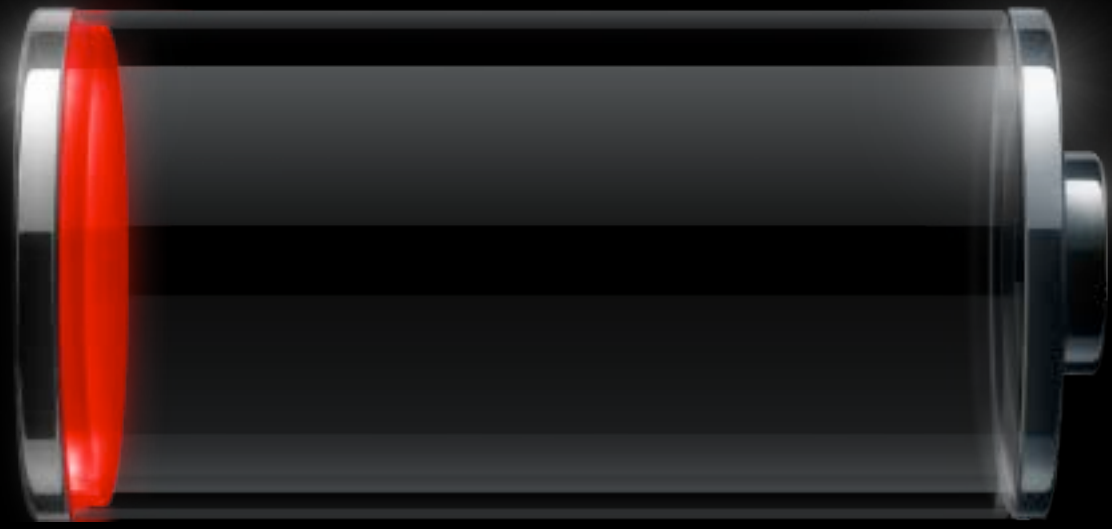






40%

Power Savings





# Idealized Fitness App

# Idealized Fitness App



— Starting Run

# Idealized Fitness App



# Idealized Fitness App



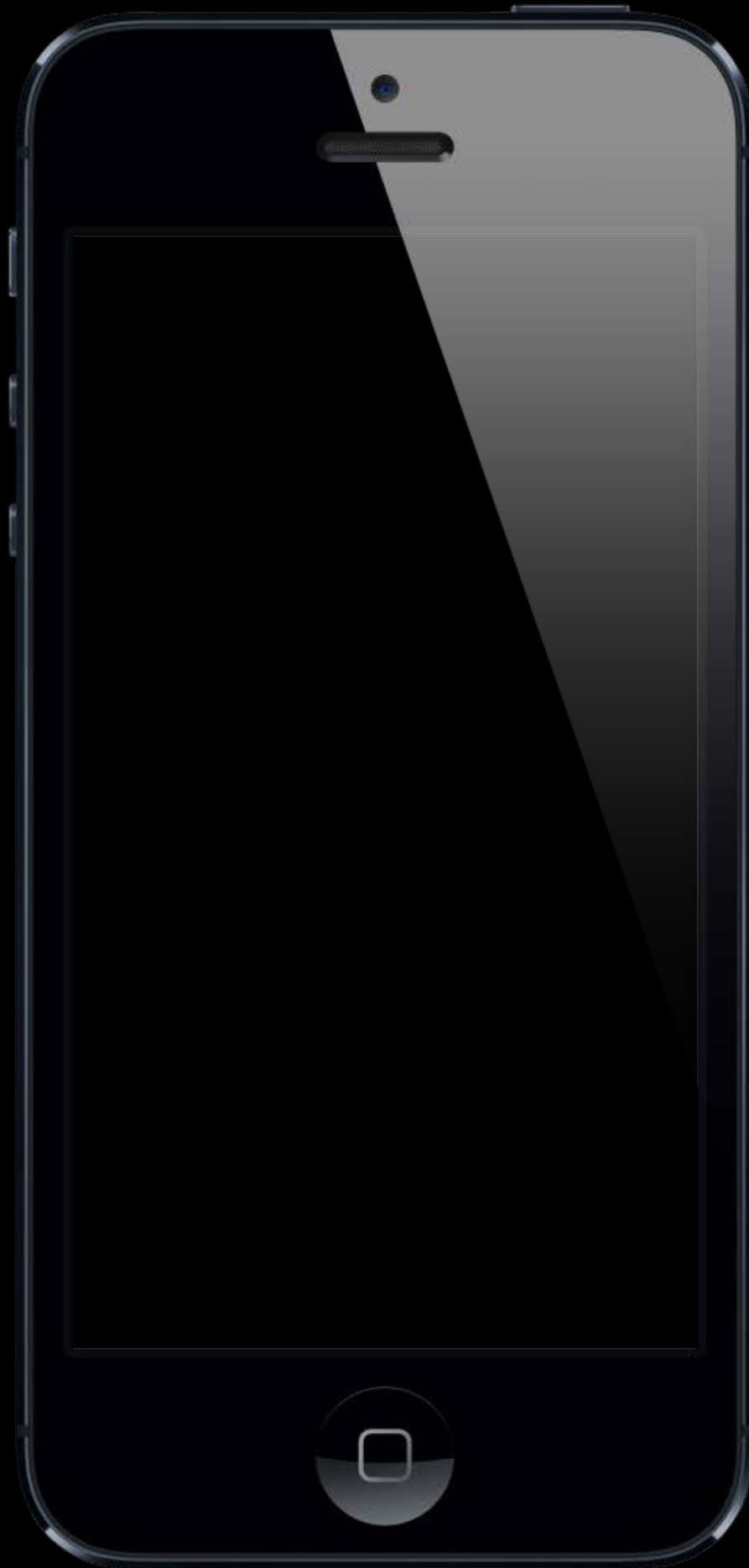
— 1 mile completed  
Total time—6 minutes

# Idealized Fitness App



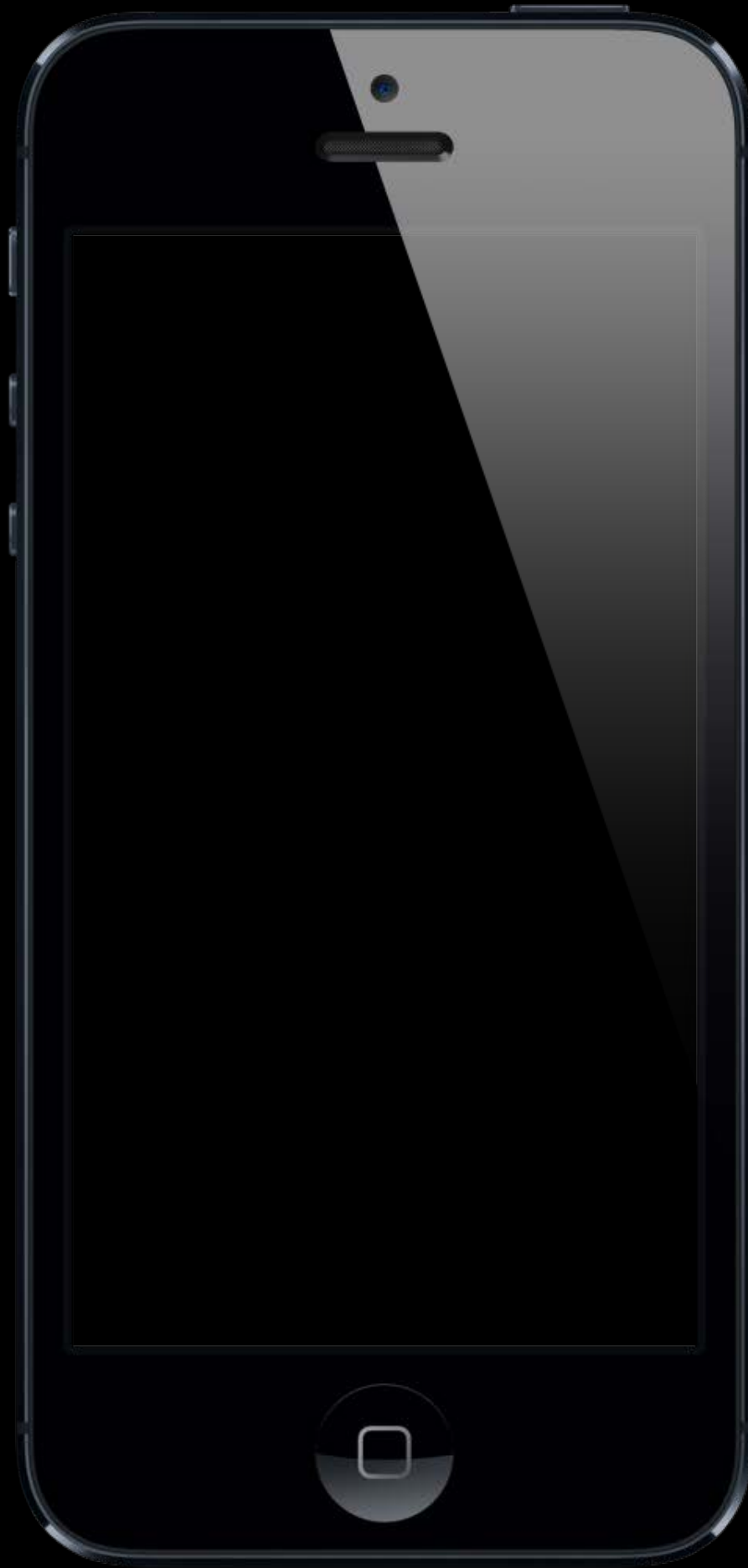


# Idealized Fitness App



— 45 minutes remaining  
Total distance—2.5 miles

# Idealized Fitness App



# Idealized Fitness App



— Workout completed  
Total distance—9.6 miles

# Deferred Location Updates

Efficient GPS usage

# Deferred Location Updates

## Efficient GPS usage

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];  
locationManager.delegate = self;  
locationManager.activityType = CLActivityTypeFitness;  
[locationManager startUpdatingLocations];
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager
    didUpdateLocations: (NSArray *) locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager
    didUpdateLocations: (NSArray *) locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```



# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                             timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager
    didUpdateLocations: (NSArray *) locations
{
    [self.run addLocations: locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible TimeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled: distance
                                                                    timeout: time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager
    didUpdateLocations: (NSArray *) locations
{
    [self.run addLocations: locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled: distance
                               timeout: time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                             timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager  
didFinishDeferredUpdatesWithError: (NSError *) error  
{  
    self.deferringUpdates = NO;  
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager  
didFinishDeferredUpdatesWithError: (NSError *) error  
{  
    self.deferringUpdates = NO;  
}
```



# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager  
didFinishDeferredUpdatesWithError:(NSError *)error  
{  
    self.deferringUpdates = NO;  
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager  
didFinishDeferredUpdatesWithError: (NSError *) error  
{  
    self.deferringUpdates = NO;  
}
```

# Deferred Location Updates

Handling errors

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredNotUpdatingLocation`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredNotUpdatingLocation`
  - `[locationManager startUpdatingLocations]`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredNotUpdatingLocation`
  - `[locationManager startUpdatingLocations]`
- `kCLErrorDeferredAccuracyTooLow`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredNotUpdatingLocation`
  - `[locationManager startUpdatingLocations]`
- `kCLErrorDeferredAccuracyTooLow`
  - `locationManager.desiredAccuracy = kCLLocationAccuracyBest`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredNotUpdatingLocation`
  - `[locationManager startUpdatingLocations]`
- `kCLErrorDeferredAccuracyTooLow`
  - `locationManager.desiredAccuracy = kCLLocationAccuracyBest`
- `kCLErrorDeferredDistanceFiltered`



# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredNotUpdatingLocation`
  - `[locationManager startUpdatingLocations]`
- `kCLErrorDeferredAccuracyTooLow`
  - `locationManager.desiredAccuracy = kCLLocationAccuracyBest`
- `kCLErrorDeferredDistanceFiltered`
  - `locationManager.distanceFilter = kCLDistanceFilterNone`

# Deferred Location Updates

Handling errors

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredCanceled`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredCanceled`
  - In response to `-disallowDeferredLocationUpdates`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredCanceled`
  - In response to `-disallowDeferredLocationUpdates`
  - In response to `-allowDeferredLocationUpdatesUntilTraveled:timeout:`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredCanceled`
  - In response to `-disallowDeferredLocationUpdates`
  - In response to `-allowDeferredLocationUpdatesUntilTraveled:timeout:`
  - API is asynchronous

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredCanceled`
  - In response to `-disallowDeferredLocationUpdates`
  - In response to `-allowDeferredLocationUpdatesUntilTraveled:timeout:`
  - API is asynchronous
  - Exactly one callback

# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```



# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

Handling errors

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`
  - Not supported

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`
  - Not supported
    - `[CLLocationManager deferredLocationUpdatesAvailable]`

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`
  - Not supported
    - `[CLLocationManager deferredLocationUpdatesAvailable]`
  - GPS is not active

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`
  - Not supported
    - `[CLLocationManager deferredLocationUpdatesAvailable]`
  - GPS is not active
    - Not ready

# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`
  - Not supported
    - `[CLLocationManager deferredLocationUpdatesAvailable]`
  - GPS is not active
    - Not ready
    - Temporary interruption



# Deferred Location Updates

## Handling errors

- `kCLErrorDeferredFailed`
  - Not supported
    - `[CLLocationManager deferredLocationUpdatesAvailable]`
  - GPS is not active
    - Not ready
    - Temporary interruption
  - Retry on next update

# Deferred Location Updates

## Efficient GPS usage

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                               timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

## Efficient GPS usage

```
- (void) locationManager: (CLLocationManager *) manager
    didUpdateLocations: (NSArray *) locations
{
    [self.run addLocations:locations];
    if (!self.deferringUpdates) {
        CLLocationDistance distance = self.run.goal - self.run.distance;
        NSTimeInterval time = [self.nextAudible timeIntervalSinceNow];
        [self.locationManager allowDeferredLocationUpdatesUntilTraveled:distance
                             timeout:time];
        self.deferringUpdates = YES;
    }
}
```

# Deferred Location Updates

Efficient GPS usage

# Deferred Location Updates

## Efficient GPS usage

- Updates delivered immediately in foreground

# Deferred Location Updates

## Efficient GPS usage

- Updates delivered immediately in foreground
- Distance travelled is optional
  - `CLLocationDistanceMax`

# Deferred Location Updates

## Efficient GPS usage

- Updates delivered immediately in foreground
- Distance travelled is optional
  - `CLLocationDistanceMax`
- Timeout is optional
  - `CLTimeIntervalMax`

# Deferred Location Updates

## Efficient GPS usage

- Updates delivered immediately in foreground
- Distance travelled is optional
  - `CLLocationDistanceMax`
- Timeout is optional
  - `CLTimeIntervalMax`
- iPhone 5



# Deferred Location Updates

## Efficient GPS usage

- Updates delivered immediately in foreground
- Distance travelled is optional
  - `CLLocationDistanceMax`
- Timeout is optional
  - `CLLocationIntervalMax`
- iPhone 5



# New Region Monitoring Features

# Selective Monitoring

# Selective Monitoring

```
- (void)monitorForArrivalAt:(CLLocation * )destination
{
    CLRegion *region = destination.region;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Selective Monitoring

```
- (void)monitorForArrivalAt:(CLPlacemark *)destination  
{  
    CLRegion *region = destination.region;  
    [self.locationManager startMonitoringForRegion:region];  
}
```

# Selective Monitoring

```
- (void)monitorForArrivalAt:(CLLocation * )destination  
{  
    CLRegion *region = destination.region;  
    [self.locationManager startMonitoringForRegion:region];  
}
```

# Selective Monitoring

```
- (void)monitorForArrivalAt:(CLLocation * )destination  
{  
    CLRegion *region = destination.region;  
    [self.locationManager startMonitoringForRegion:region];  
}
```

# Selective Monitoring



```
- (void)monitorForArrivalAt:(CLLocation * )destination
{
    CLRegion *region = destination.region;

    [self.locationManager startMonitoringForRegion:region];
}
```



# Selective Monitoring



```
- (void)monitorForArrivalAt:(CLLocationMark *)destination
{
    CLRegion *region = destination.region;
    region.notifyOnEntry = YES;

    [self.locationManager startMonitoringForRegion:region];
}
```

# Selective Monitoring



```
- (void)monitorForArrivalAt:(CLLocationMark *)destination
{
    CLRegion *region = destination.region;
    region.notifyOnEntry = YES;
    region.notifyOnExit = NO;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Selective Monitoring

CLRegion



# Selective Monitoring

## CLRegion

- Entry

```
@property(nonatomic, assign) BOOL notifyOnEntry;
```



# Selective Monitoring

## CLRegion



- Entry

```
@property(nonatomic, assign) BOOL notifyOnEntry;
```

- Exit

```
@property(nonatomic, assign) BOOL notifyOnExit;
```

# Selective Monitoring

## CLRegion



- Entry

```
@property(nonatomic, assign) BOOL notifyOnEntry;
```

- Exit

```
@property(nonatomic, assign) BOOL notifyOnExit;
```

- Default to YES

Region State?

# Region State?

```
- (void)monitorForDeparture
```

```
{
```

```
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
```

```
    CLRegion *region = [CLRegion initWithCenter:where  
                                                                radius:100.0 // meters  
                                                                identifier:@"Departure"];
```

```
    self.notifyOnEntry = NO;
```

```
    self.notifyOnExit = YES;
```

```
    [self.locationManager startMonitoringForRegion:region];
```

```
}
```



# Region State?

```
- (void)monitorForDeparture
{
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
    CLRegion *region = [CLRegion initWithCenter:where
                                   radius:100.0 // meters
                                   identifier:@"Departure"];

    self.notifyOnEntry = NO;
    self.notifyOnExit = YES;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Region State?

```
- (void)monitorForDeparture
{
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
    CLRegion *region = [CLRegion initWithCenter:where
                                   radius:100.0 // meters
                                   identifier:@"Departure"];
    self.notifyOnEntry = NO;
    self.notifyOnExit = YES;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Region State?

```
- (void)monitorForDeparture
{
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
    CLRegion *region = [CLRegion initWithCenter:where
                                   radius:100.0 // meters
                                   identifier:@"Departure"];
    self.notifyOnEntry = NO;
    self.notifyOnExit = YES;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Region State?

```
- (void)monitorForDeparture
{
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
    CLRegion *region = [CLRegion initWithCenter:where
                                   radius:100.0 // meters
                                   identifier:@"Departure"];

    self.notifyOnEntry = NO;
    self.notifyOnExit = YES;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Region State?

```
- (void)monitorForDeparture
{
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
    CLRegion *region = [CLRegion initWithCenter:where
                                   radius:100.0 // meters
                                   identifier:@"Departure"];

    self.notifyOnEntry = NO;
    self.notifyOnExit = YES;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Region State?

```
- (void)monitorForDeparture
{
    CLLocationCoordinate2D where = self.lastLocation.coordinate;
    CLRegion *region = [CLRegion initWithCenter:where
                                   radius:100.0 // meters
                                   identifier:@"Departure"];

    self.notifyOnEntry = NO;
    self.notifyOnExit = YES;
    [self.locationManager startMonitoringForRegion:region];
}
```

# Region State?



```
- (void) locationManager: (CLLocationManager *) manager
    didDetermineState: (CLRegionState) state
    forRegion: (CLRegion *) region
{
    if ([region isEqualToString:@"Departure"]) {
        if (CLRegionStateOutside == state) {
            // TODO: notify user of departure
        }
    }
}
```

# Region State?



```
- (void) locationManager: (CLLocationManager *) manager
    didDetermineState: (CLRegionState) state
    forRegion: (CLRegion *) region
{
    if ([region isEqualToString:@"Departure"]) {
        if (CLRegionStateOutside == state) {
            // TODO: notify user of departure
        }
    }
}
```



# Region State?



```
- (void) locationManager: (CLLocationManager *) manager
    didDetermineState: (CLRegionState) state
    forRegion: (CLRegion *) region
{
    if ([region isEqualToString:@"Departure"]) {
        if (CLRegionStateOutside == state) {
            // TODO: notify user of departure
        }
    }
}
```

# Region State?



```
- (void) locationManager: (CLLocationManager *) manager
    didDetermineState: (CLRegionState) state
    forRegion: (CLRegion *) region
{
    if ([region isEqualToString:@"Departure"]) {
        if (CLRegionStateOutside == state) {
            // TODO: notify user of departure
        }
    }
}
```

# Region State?



```
- (void) locationManager: (CLLocationManager *) manager
    didDetermineState: (CLRegionState) state
    forRegion: (CLRegion *) region
{
    if ([region isEqualToString:@"Departure"]) {
        if (CLRegionStateOutside == state) {
            // TODO: notify user of departure
        }
    }
}
```

# Region State?



# Region State?



- Delegate

- (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    `forRegion:(CLRegion *)region;`

# Region State?



- Delegate

- (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    `forRegion:(CLRegion *)region;`

- Invoked while running

# Region State?



- Delegate
  - (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    `forRegion:(CLRegion *)region;`
    - Invoked while running
- Three possible states

# Region State?



- Delegate

- (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    forRegion:(CLRegion \*)region;

- Invoked while running

- Three possible states

- `CLRegionStateInside`



# Region State?



- Delegate

- (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    forRegion:(CLRegion \*)region;

- Invoked while running

- Three possible states

- `CLRegionStateInside`

- `CLRegionStateOutside`

# Region State?



- Delegate

- (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    forRegion:(CLRegion \*)region;

- Invoked while running

- Three possible states

- `CLRegionStateInside`
  - `CLRegionStateOutside`
  - `CLRegionStateUnknown`

# Region State?



- Delegate

- (void) `locationManager:(CLLocationManager *)manager`  
    `didDetermineState:(CLRegionState)state`  
    forRegion:(CLRegion \*)region;

- Invoked while running

- Three possible states

- `CLRegionStateInside`  
`CLRegionStateOutside`  
`CLRegionStateUnknown`

- Queryable

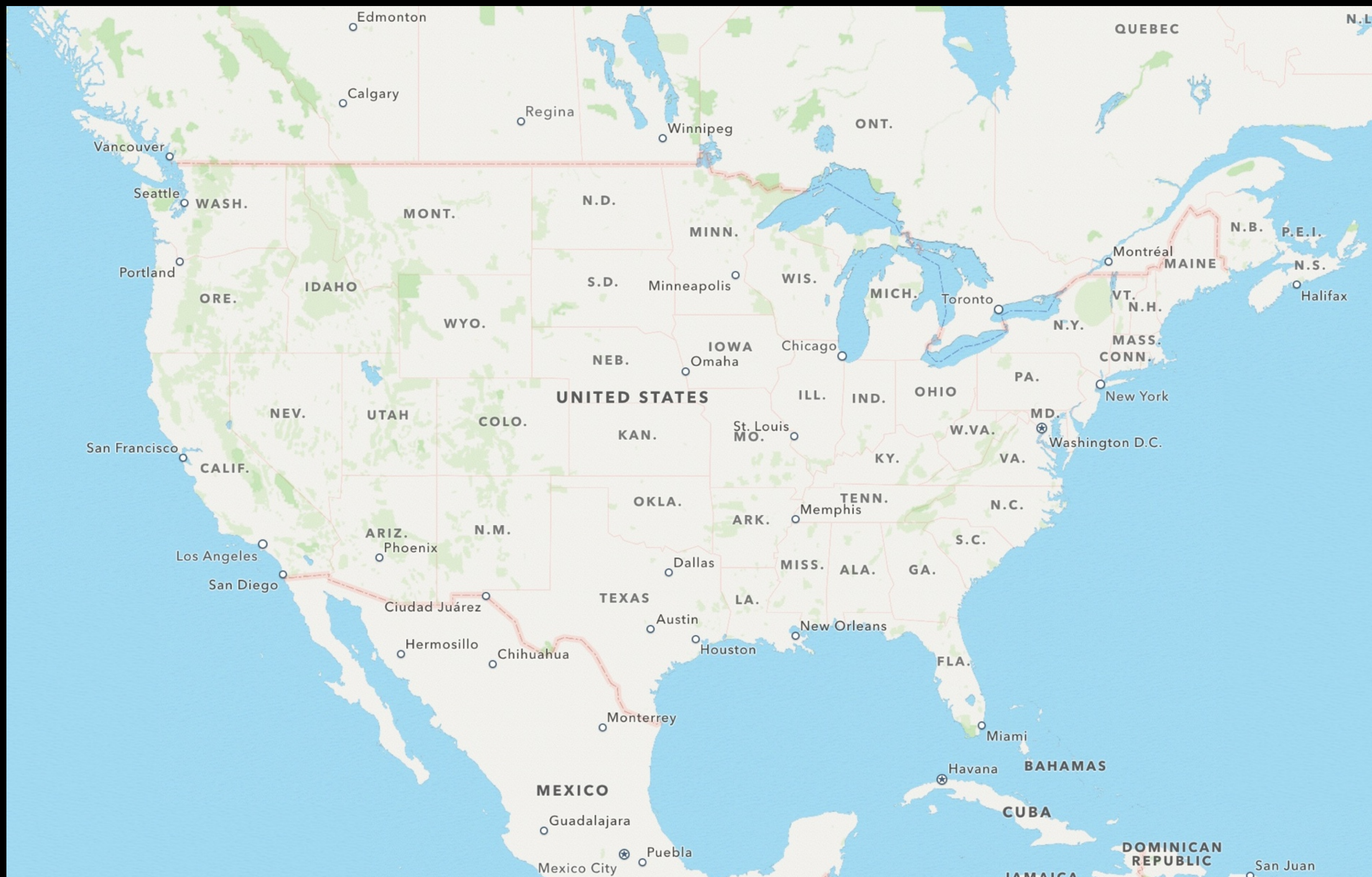
- (void) `requestStateForRegion:(CLRegion *)region;`

# Jay's Donut Shop

Case study



# Jay's Donut Shop





# Jay's Donut Shop





Jay's Donut Shop.app

# Jay's Donut Shop.app





# Jay's Donut Shop.app

Welcome customers

# Jay's Donut Shop.app

Welcome customers

```
[locationManager startMonitoringForRegion:cupertino];
```

# Jay's Donut Shop.app

## Welcome customers

```
[locationManager startMonitoringForRegion:cupertino];  
[locationManager startMonitoringForRegion:sanFrancisco];  
[locationManager startMonitoringForRegion:sanFranciscoMoscone];  
[locationManager startMonitoringForRegion:sanFranciscoTwinPeaks];  
[locationManager startMonitoringForRegion:sanJose];  
[locationManager startMonitoringForRegion:losAngeles];  
[locationManager startMonitoringForRegion:newYork];  
[locationManager startMonitoringForRegion:denver];  
[locationManager startMonitoringForRegion:phoenix];  
[locationManager startMonitoringForRegion:seattle];  
[locationManager startMonitoringForRegion:chicagoLoop];  
[locationManager startMonitoringForRegion:chicagoBelmont];  
[locationManager startMonitoringForRegion:washingtonDC];  
[locationManager startMonitoringForRegion:toronto];  
...
```

# Jay's Donut Shop.app

Welcome customers

# Jay's Donut Shop.app

Welcome customers

# Jay's Donut Shop.app

Welcome customers

```
[locationManager startMonitoringForRegion:  ];
```





# Jay's Donut Shop

## Physical





# Jay's Donut Shop

## Virtual





# Jay's Donut Shop

## Virtual





# Jay's Donut Shop

## Virtual



Jay's Donut Shop





# Jay's Donut Shop

## Virtual



Jay's Donut Shop

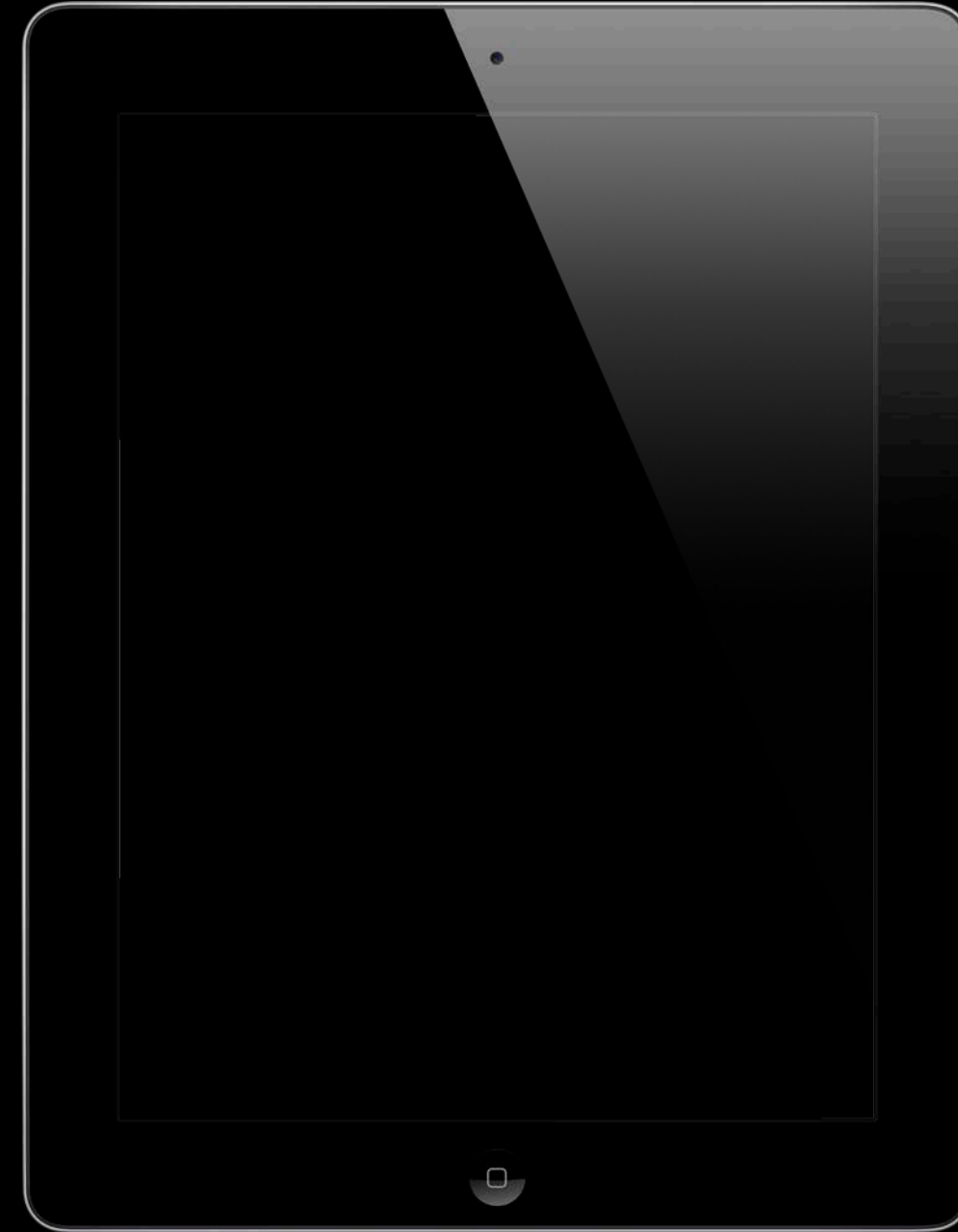


Cupertino

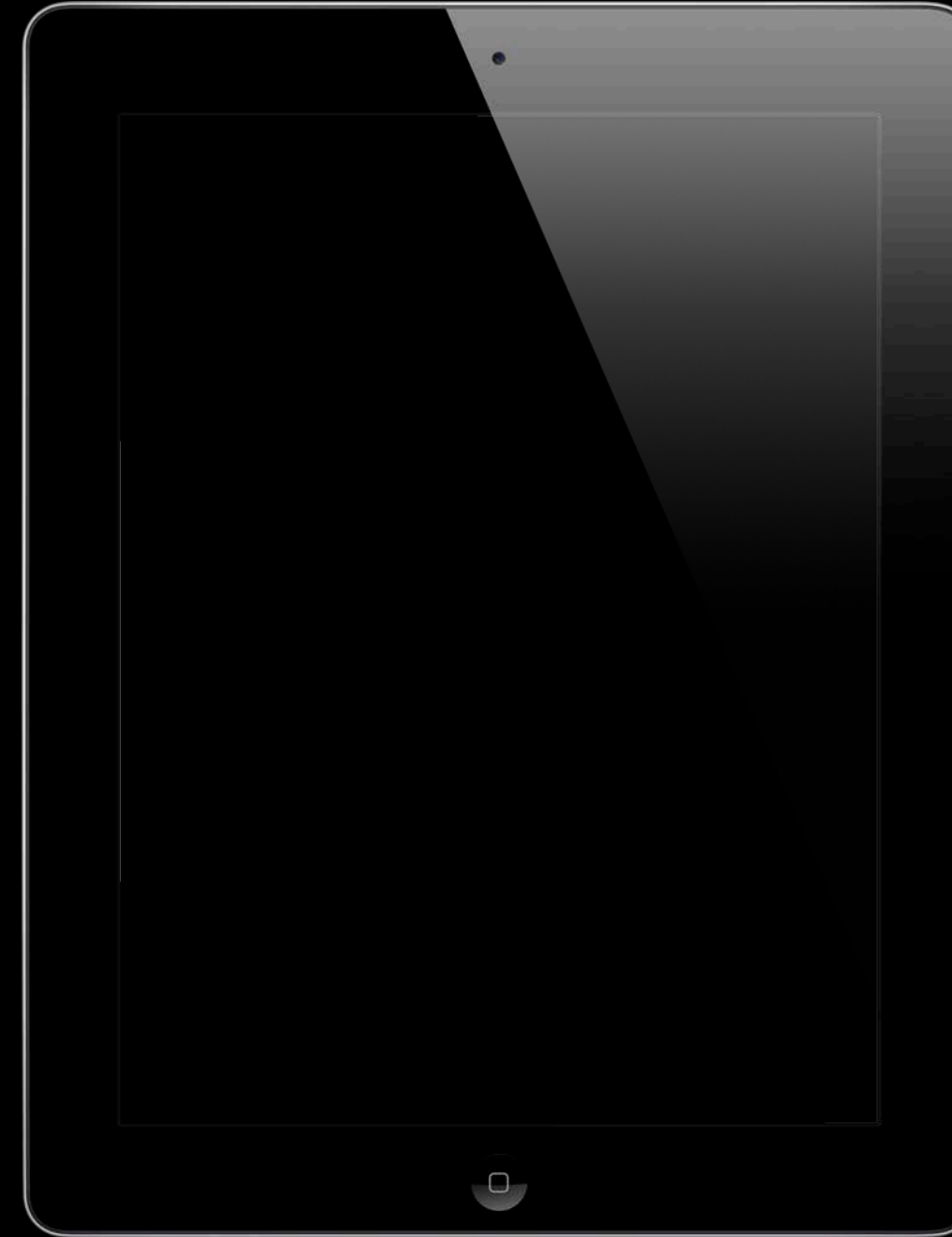


# iBeacons

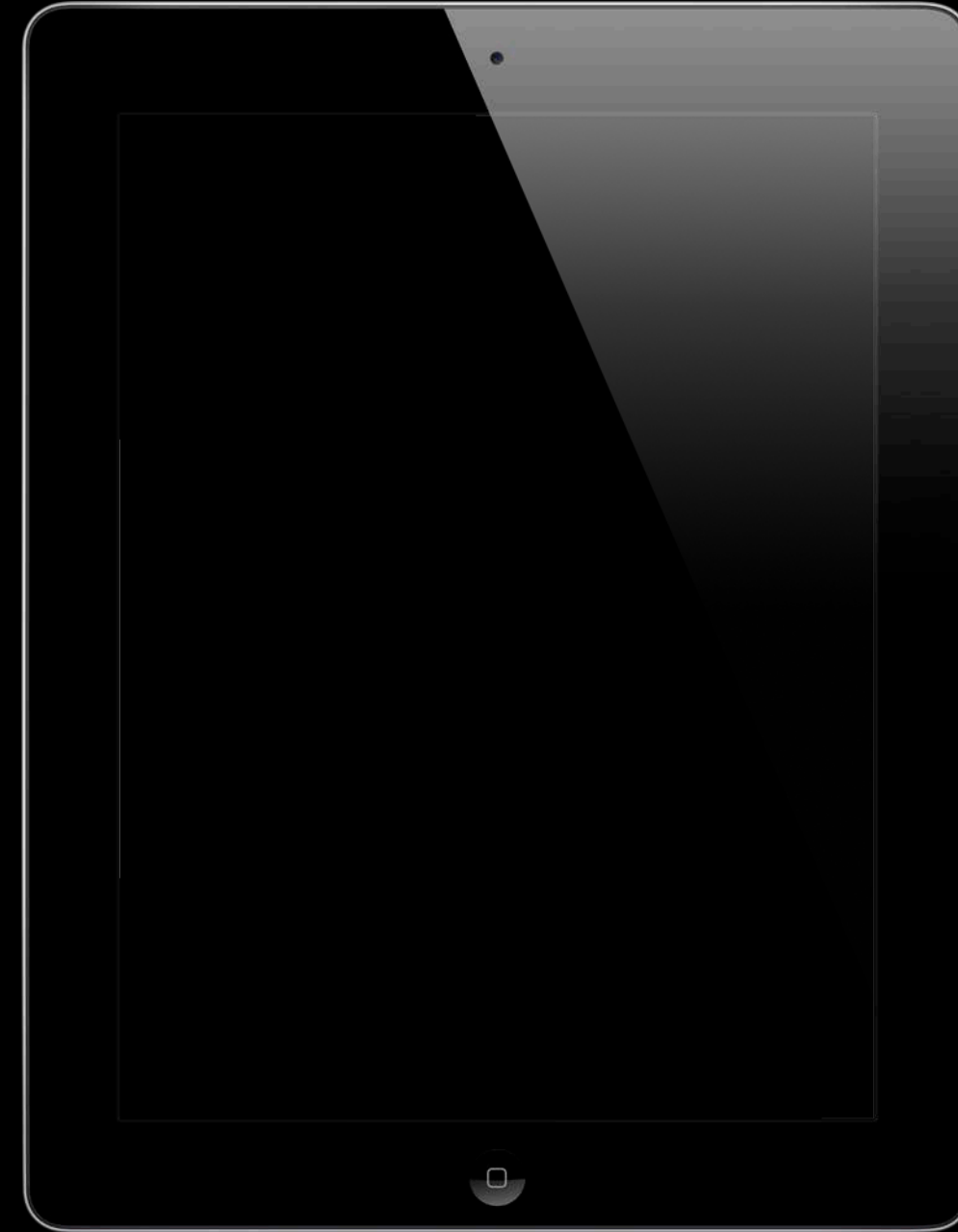
# iBeacons



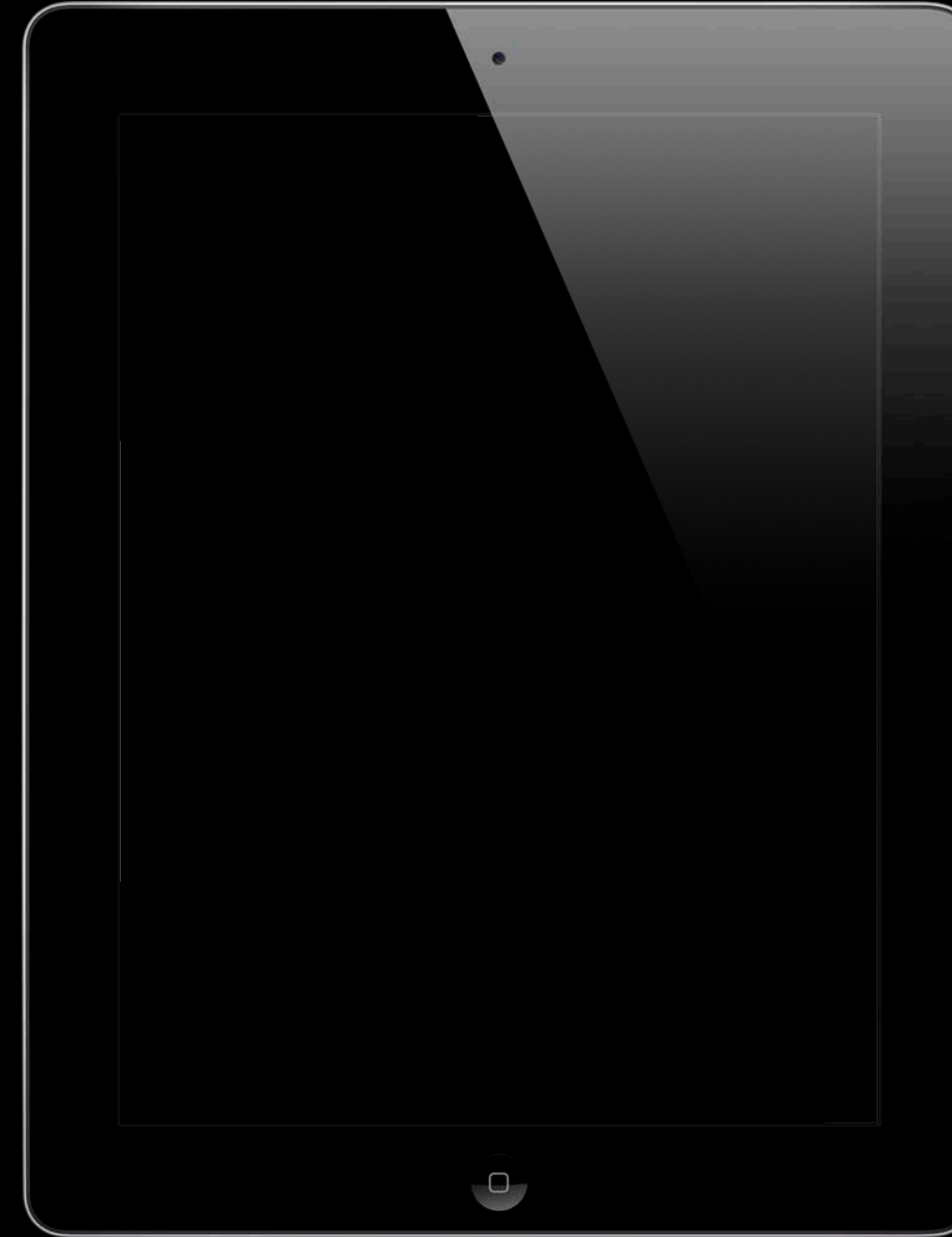
# iBeacons



# iBeacons

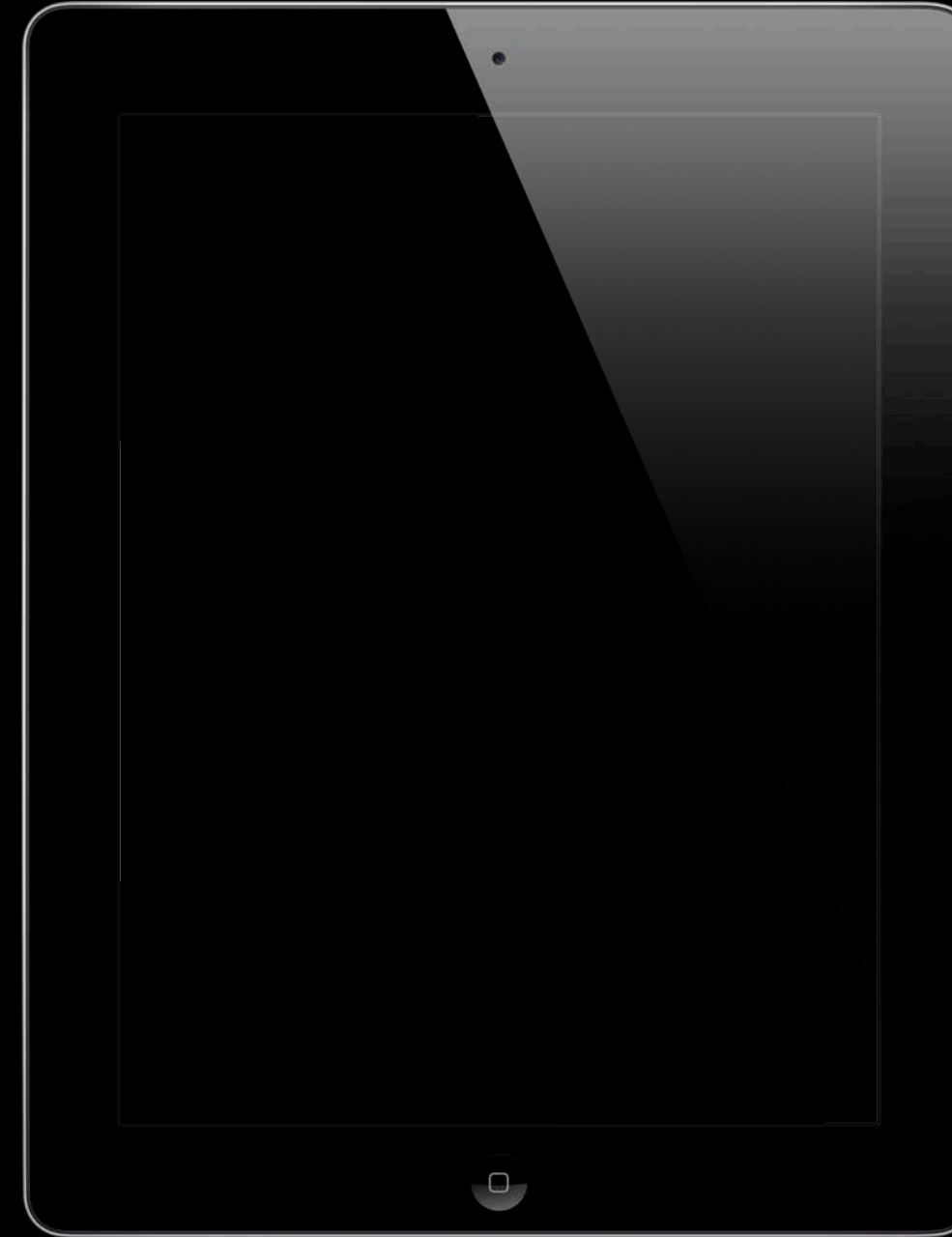


# iBeacons

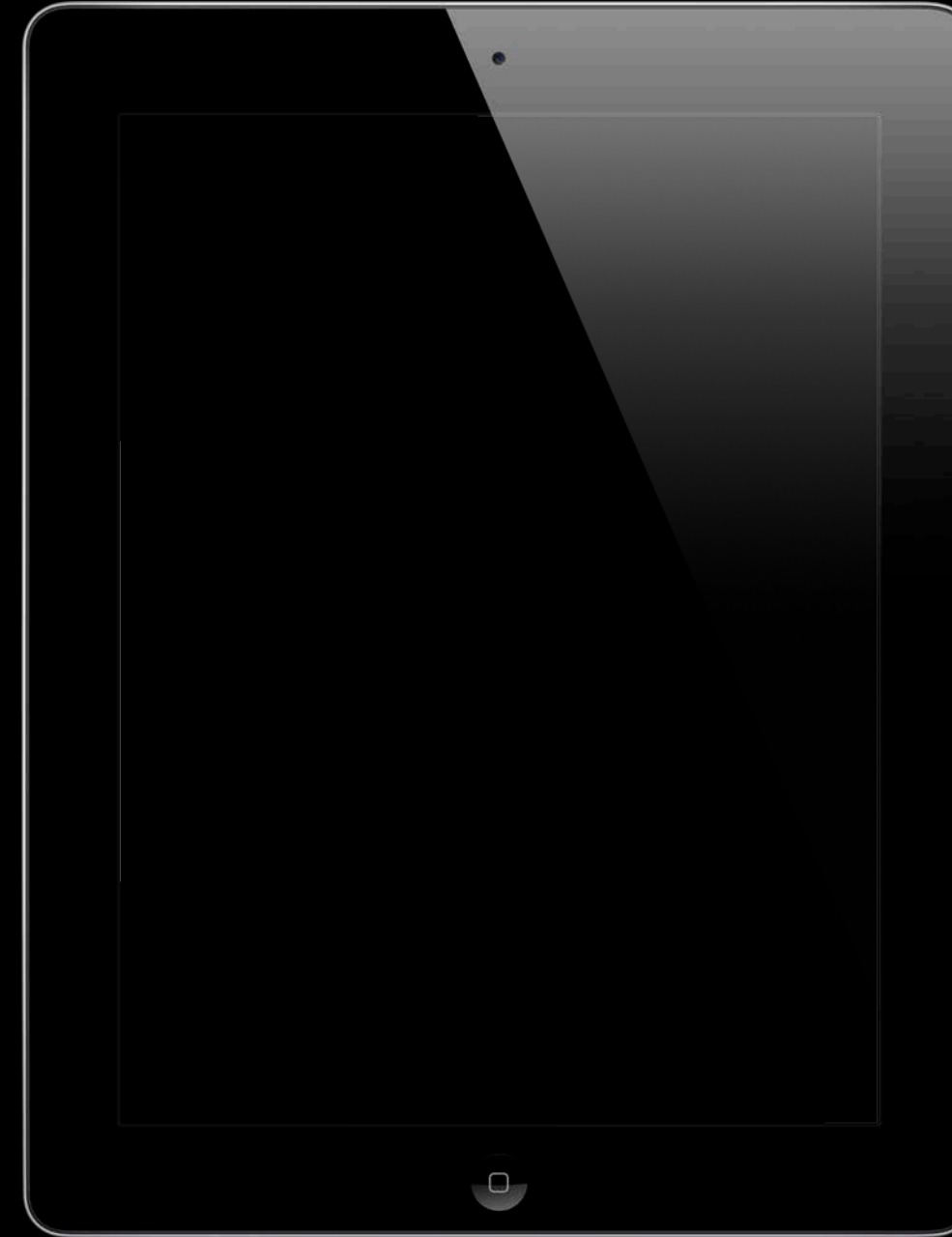




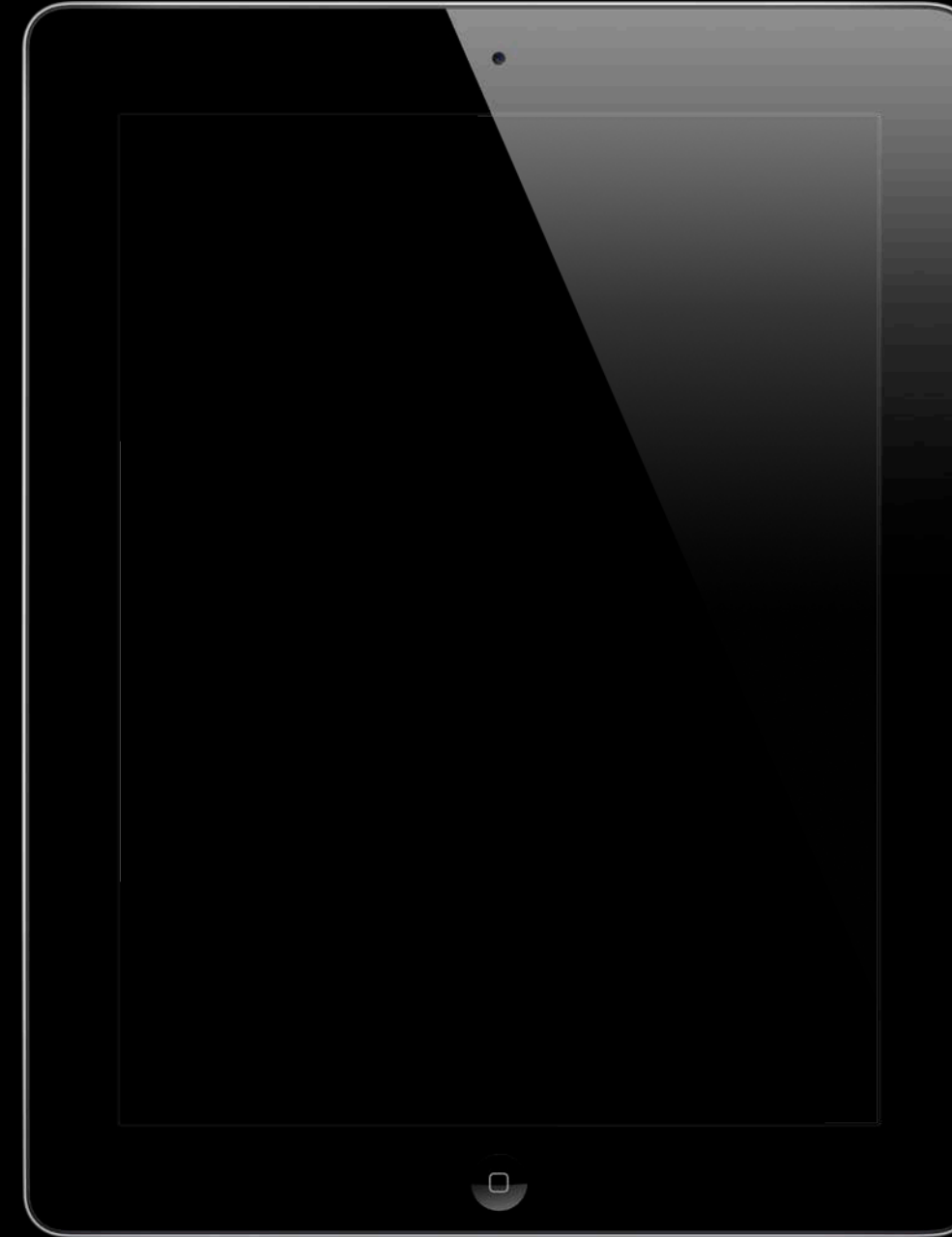
# iBeacons



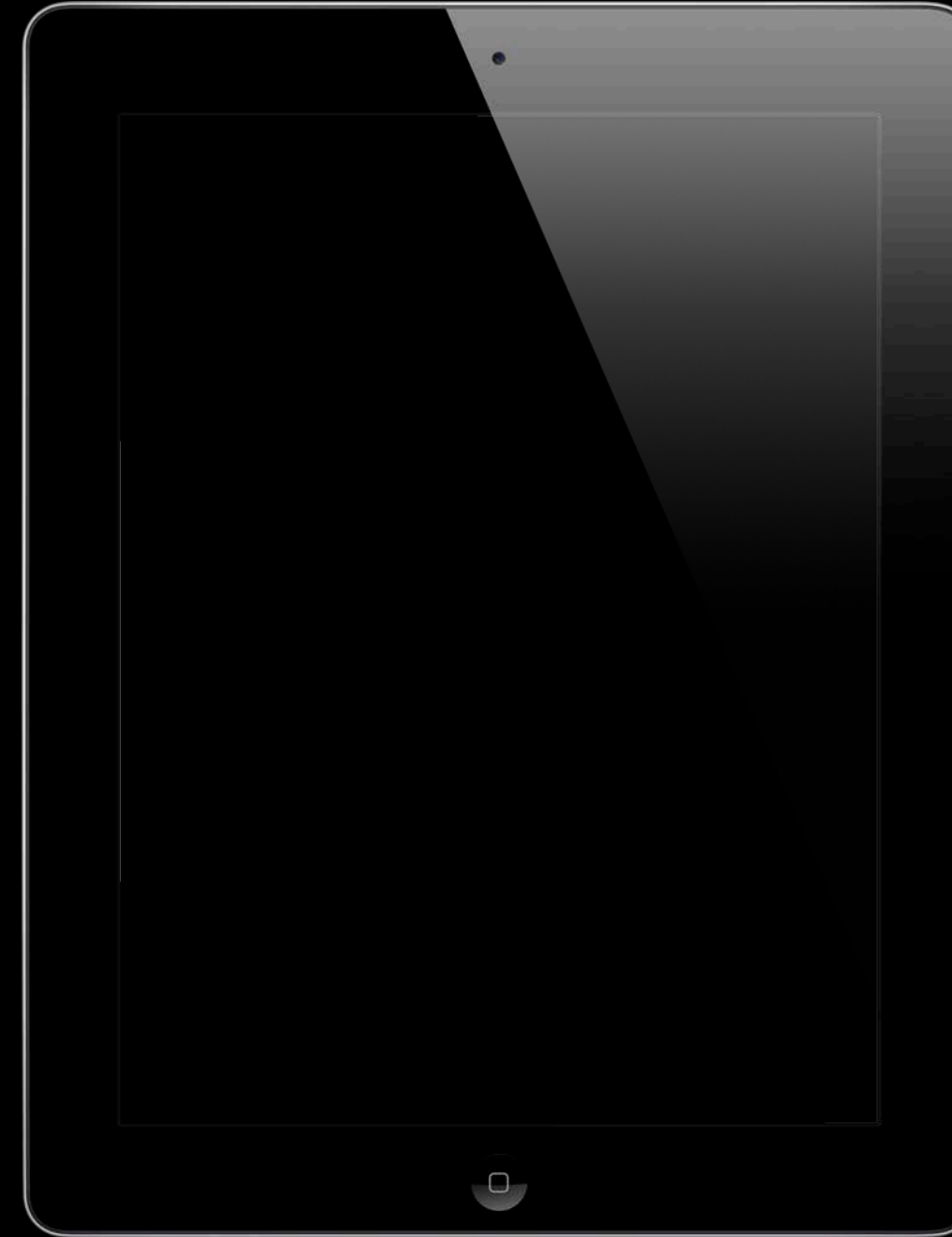
# iBeacons



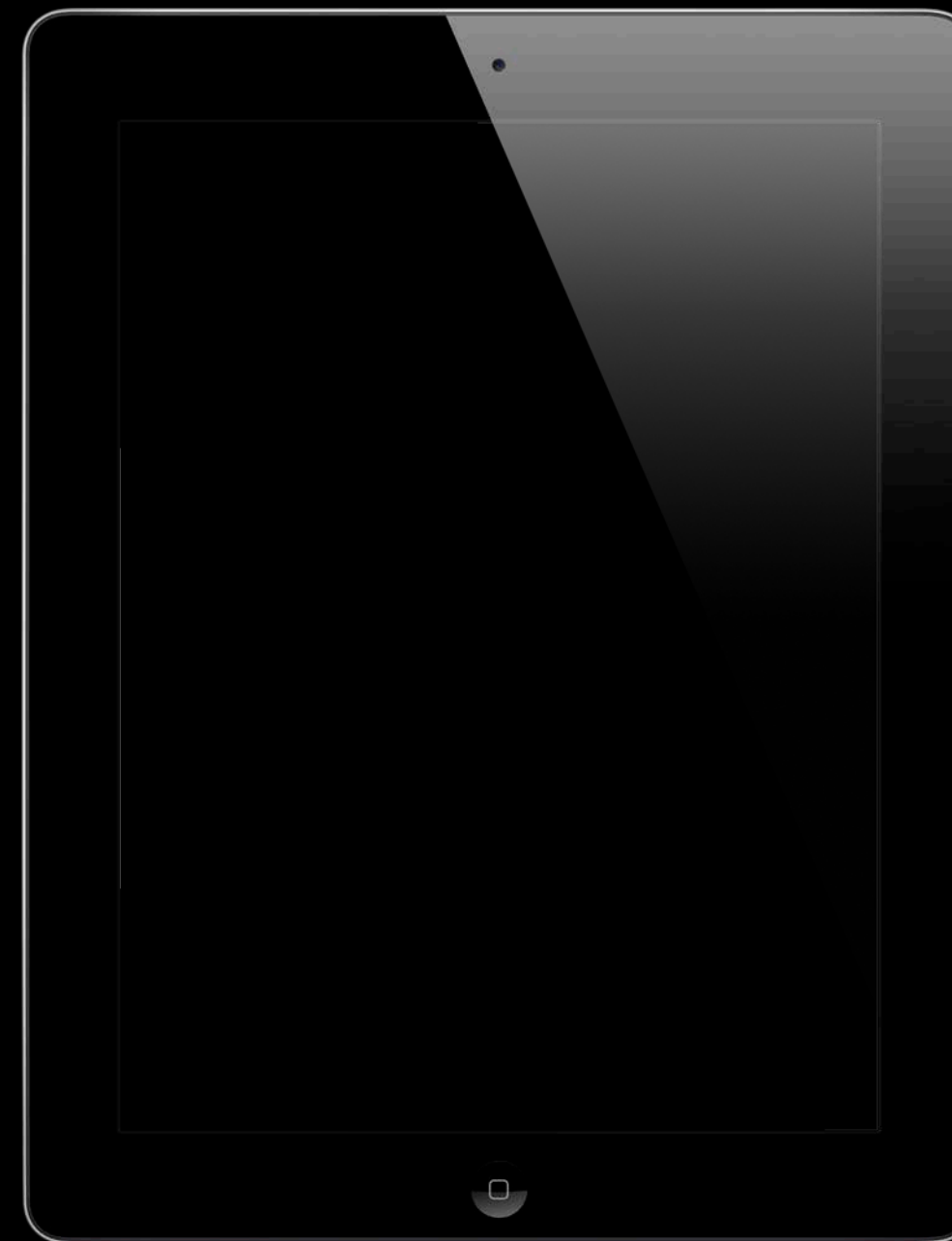
# iBeacons



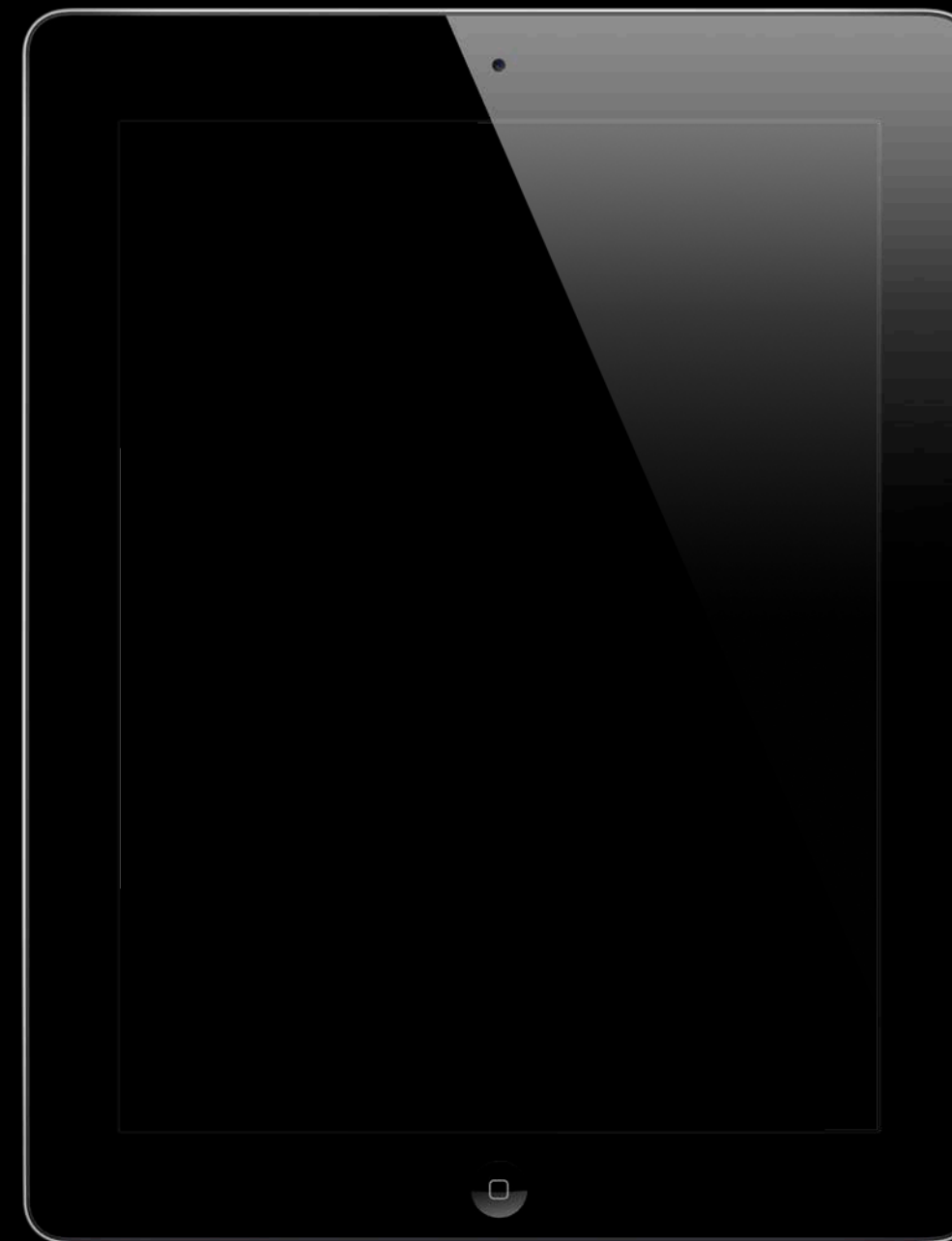
# iBeacons



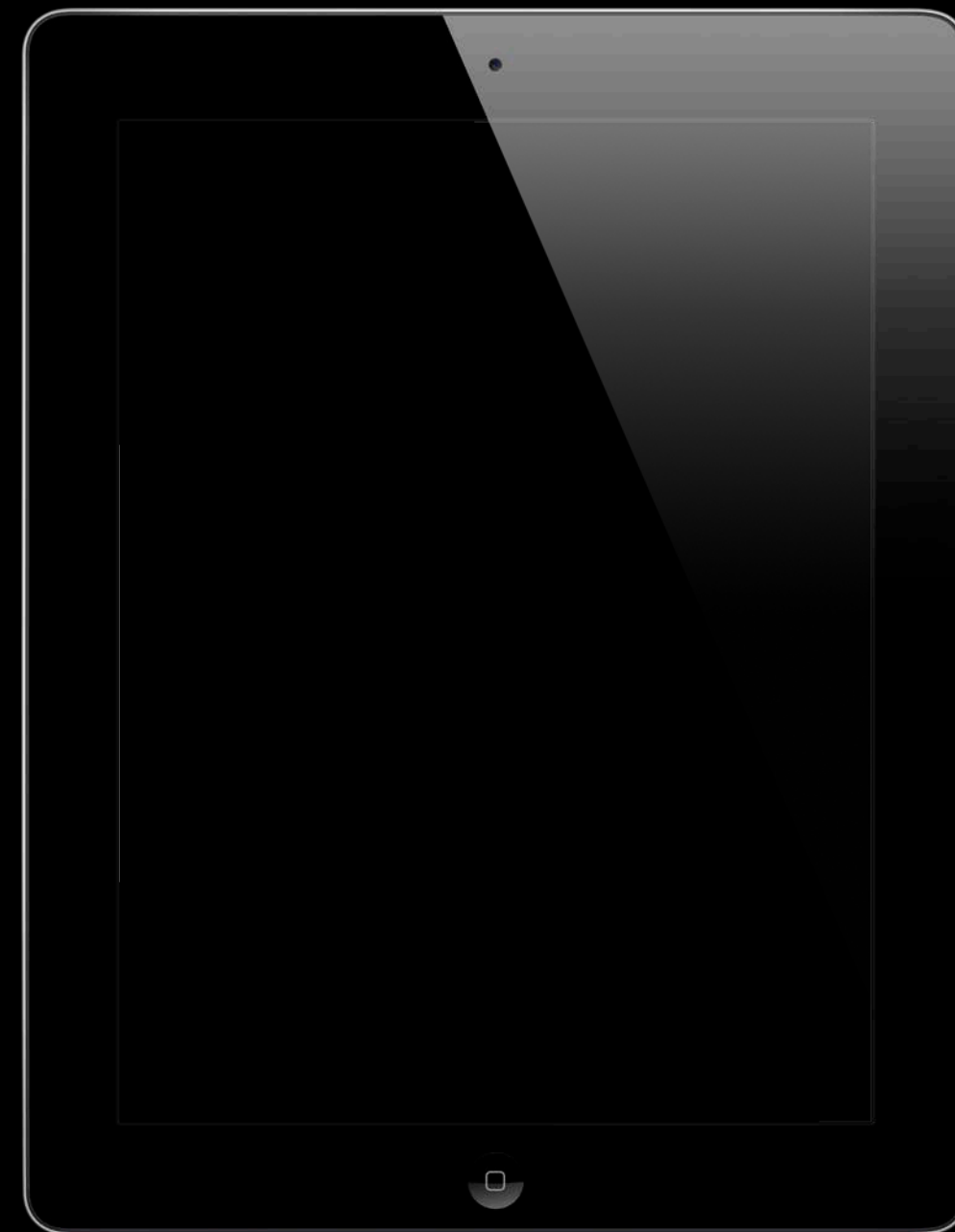
# iBeacons



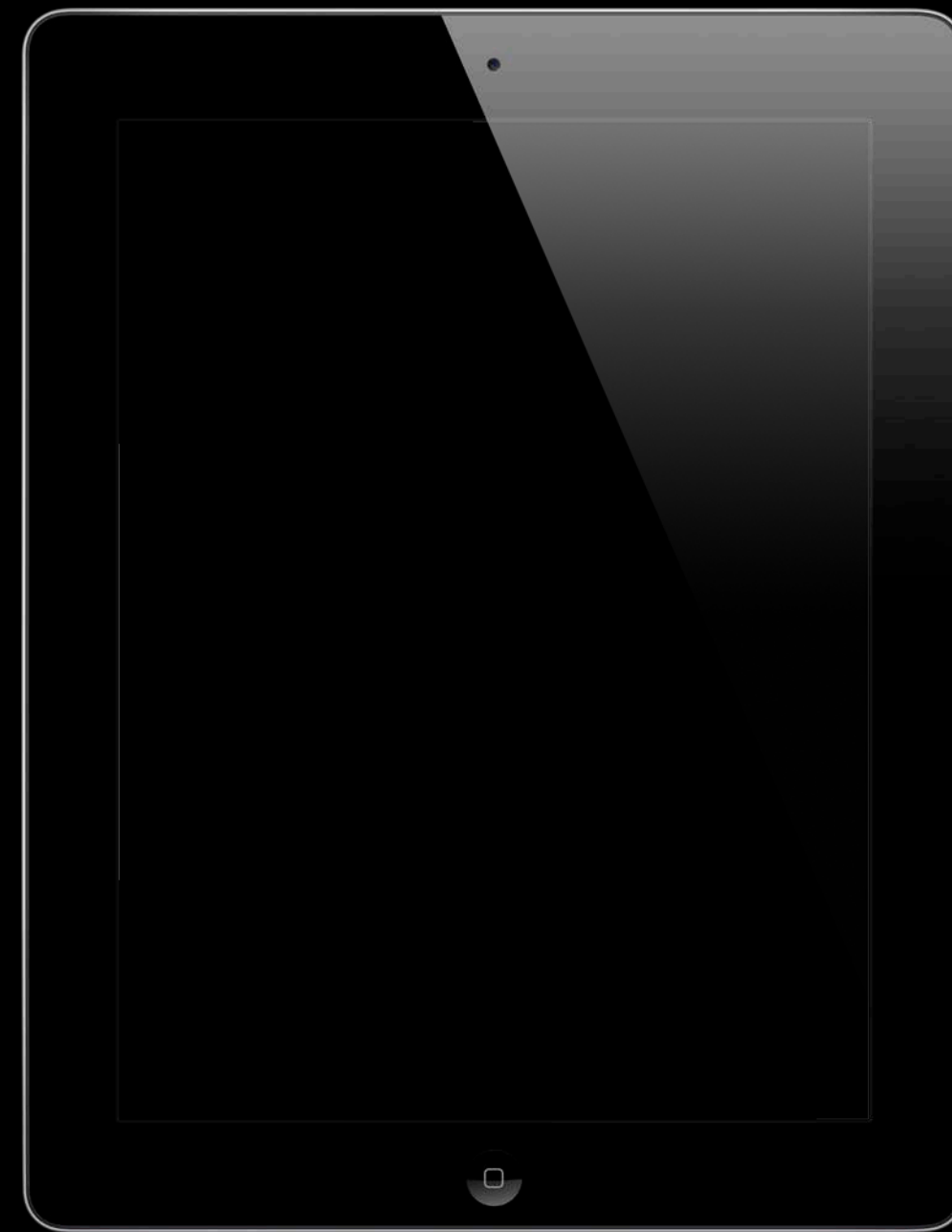
# iBeacons



# iBeacons

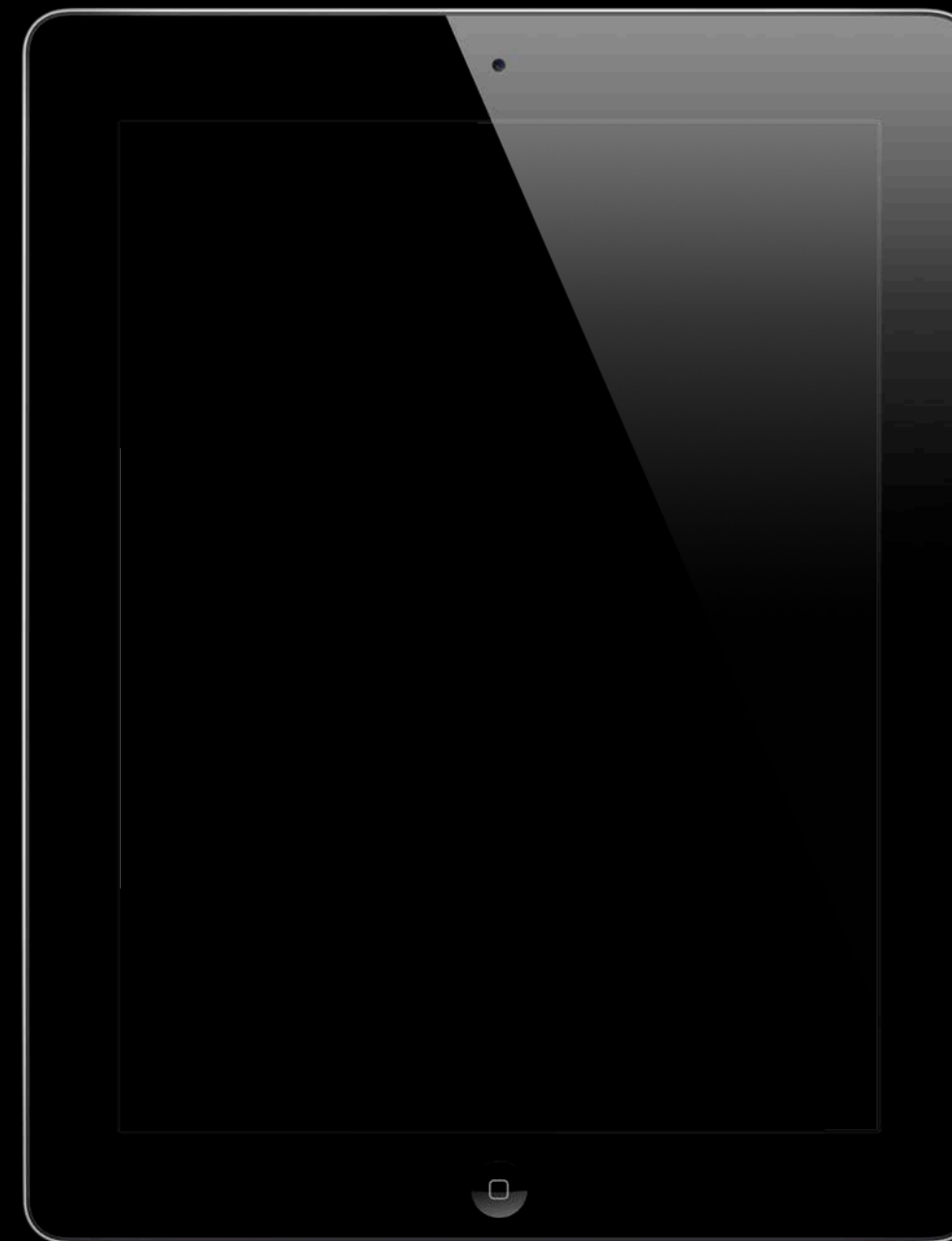


# iBeacons

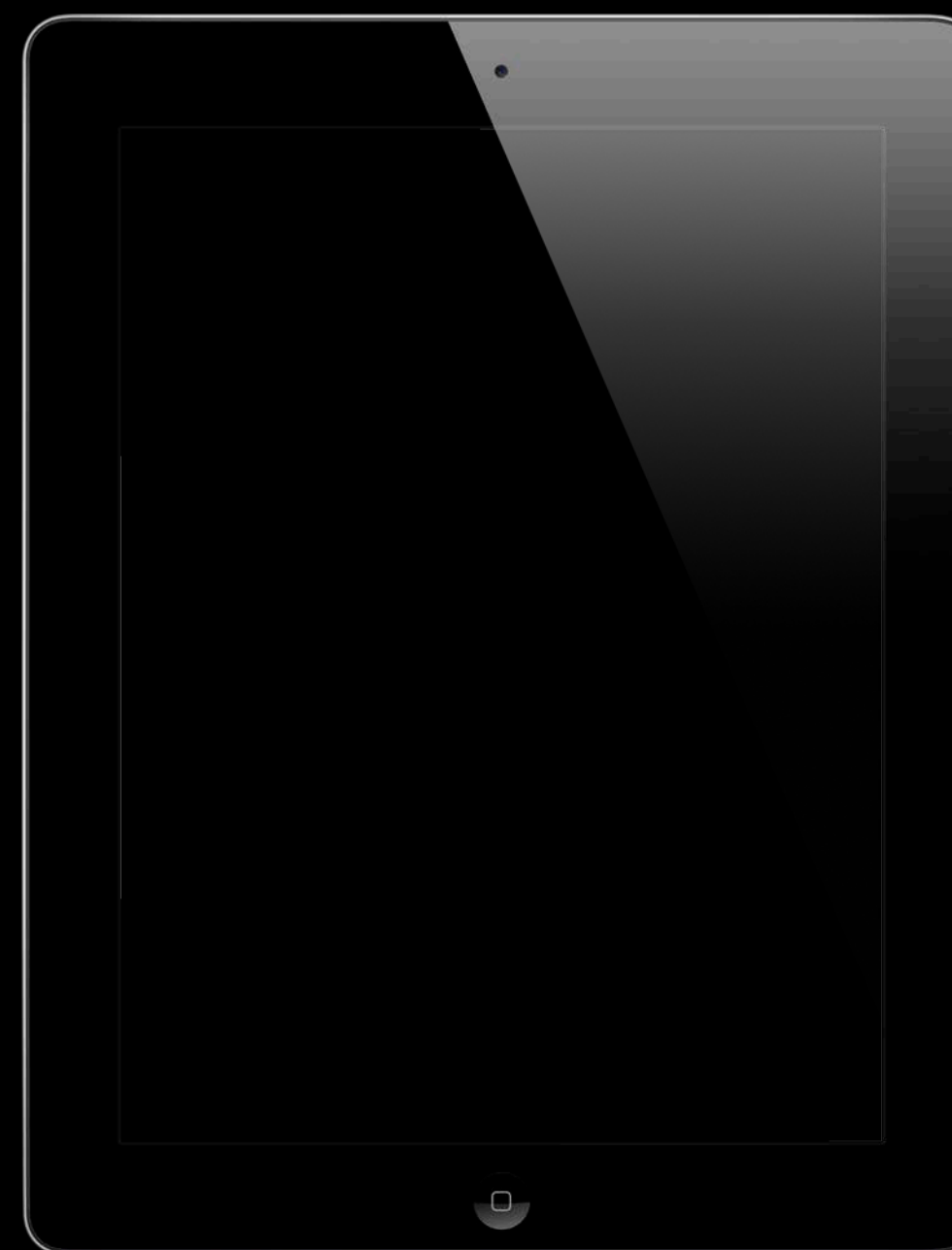




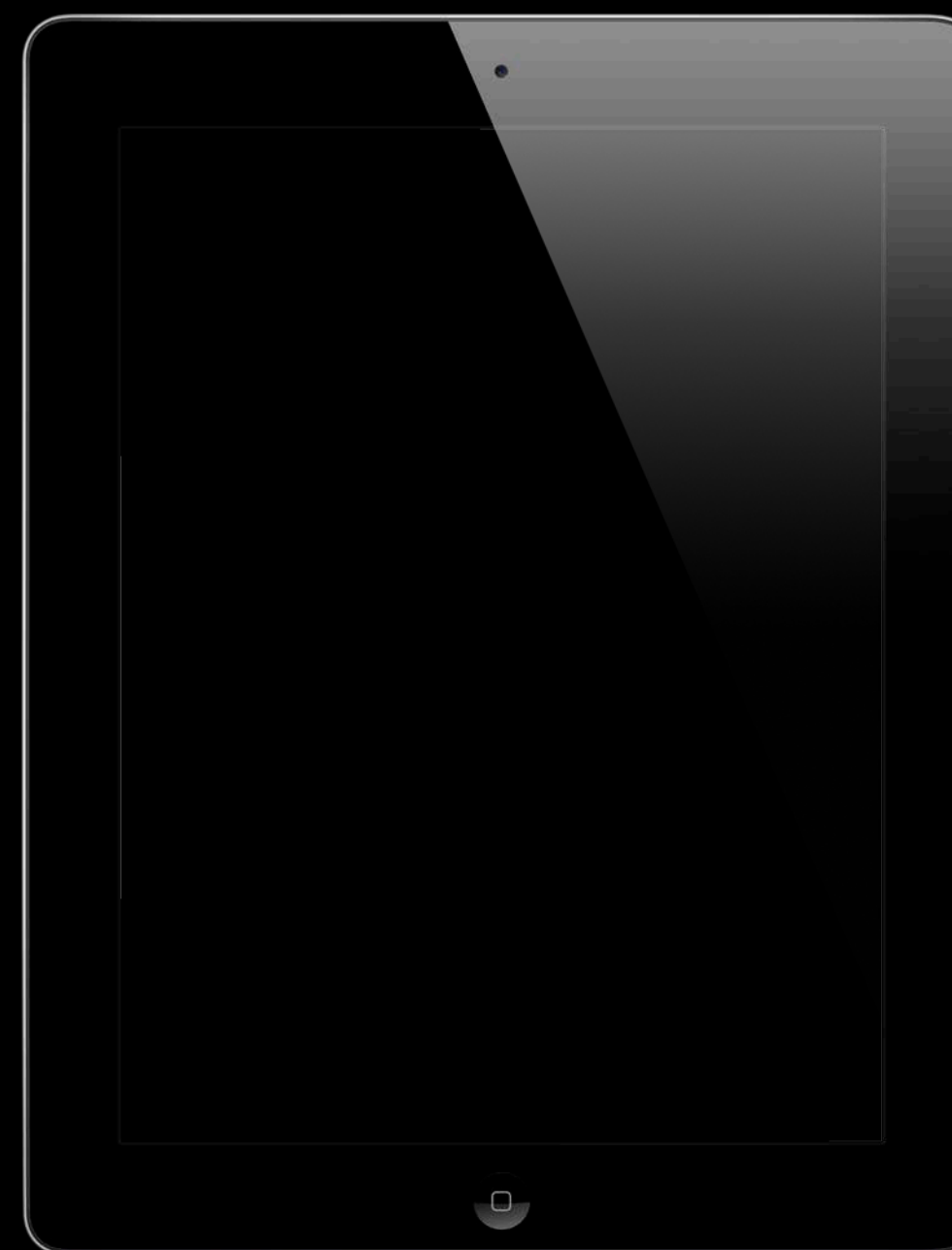
# iBeacons



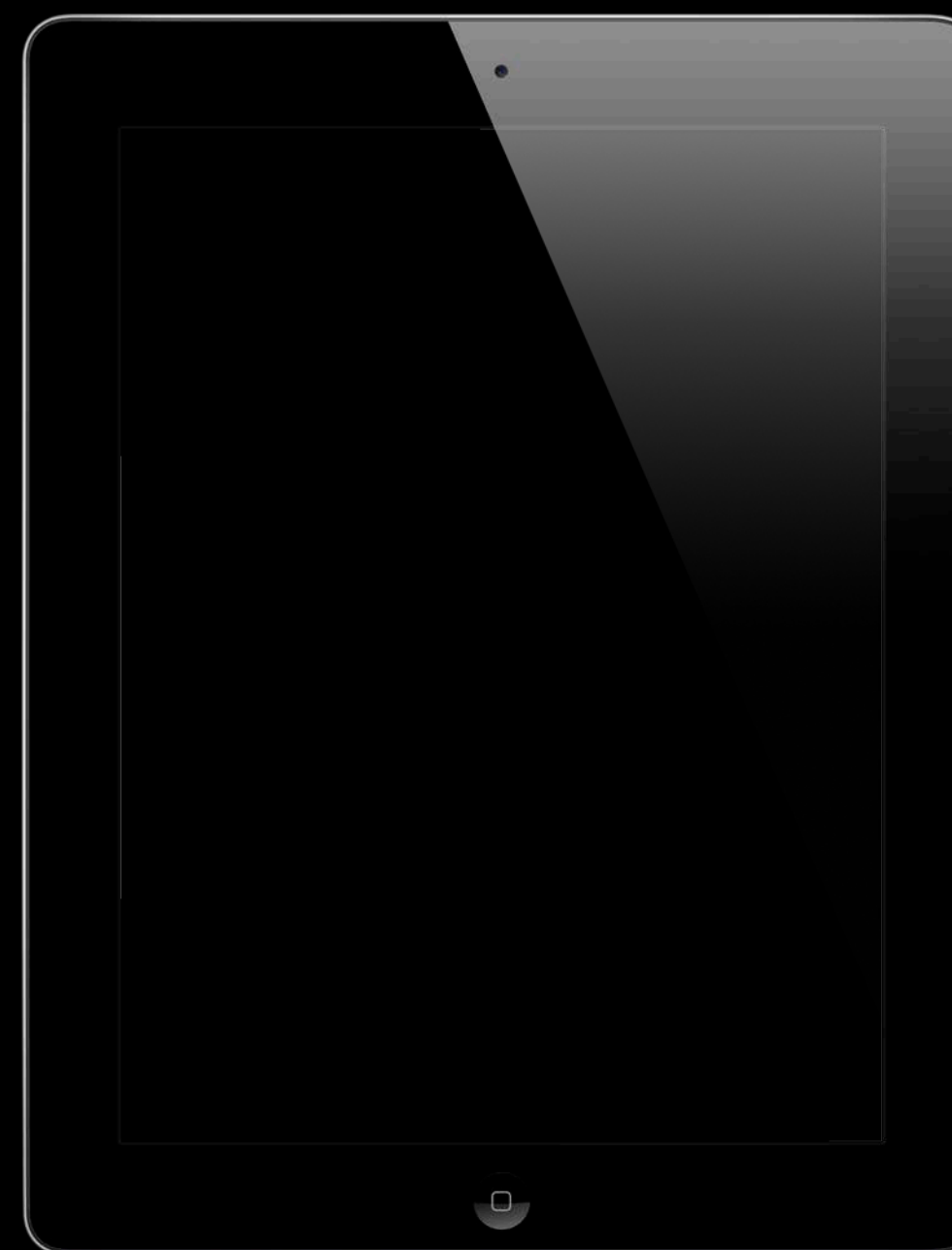
# iBeacons



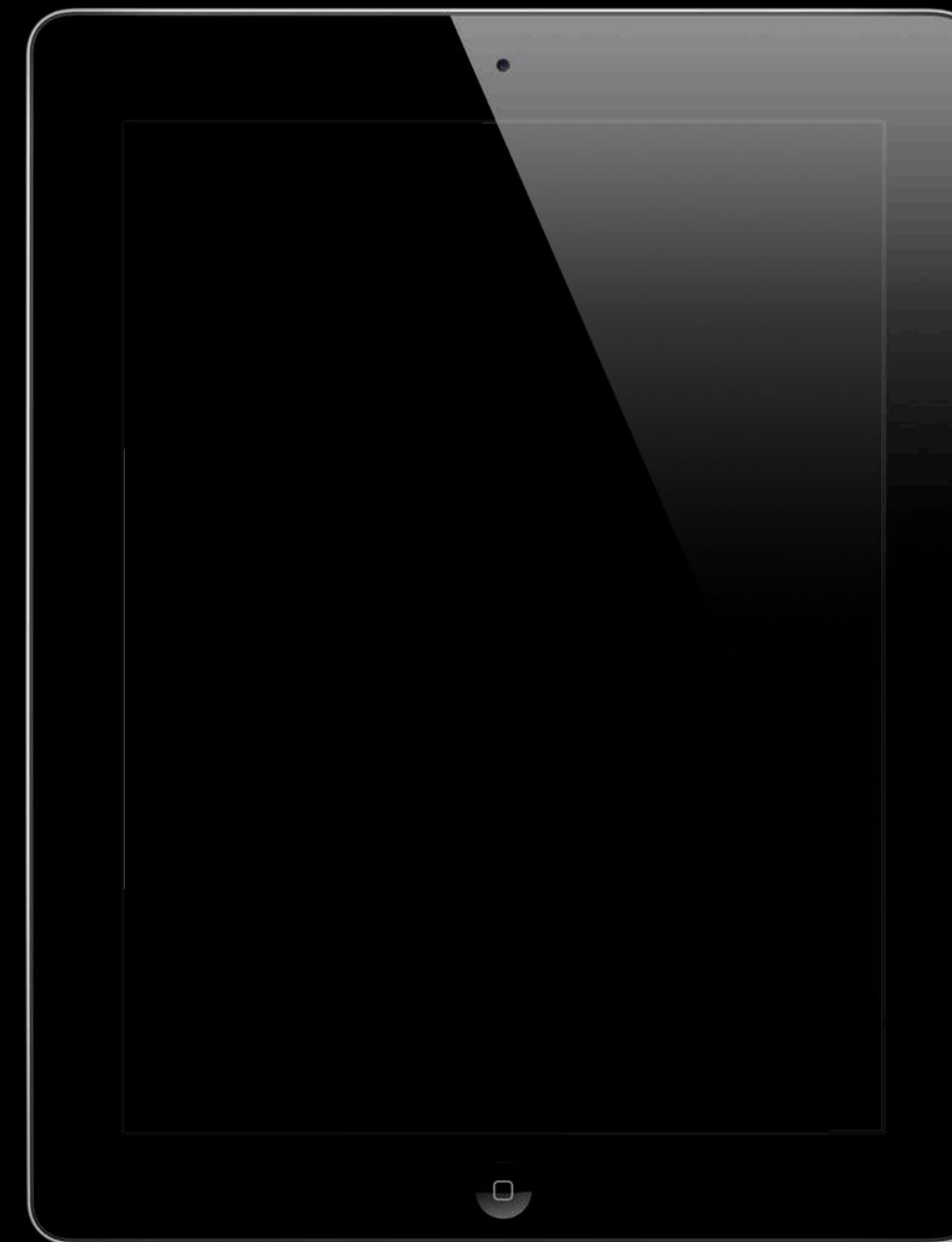
# iBeacons



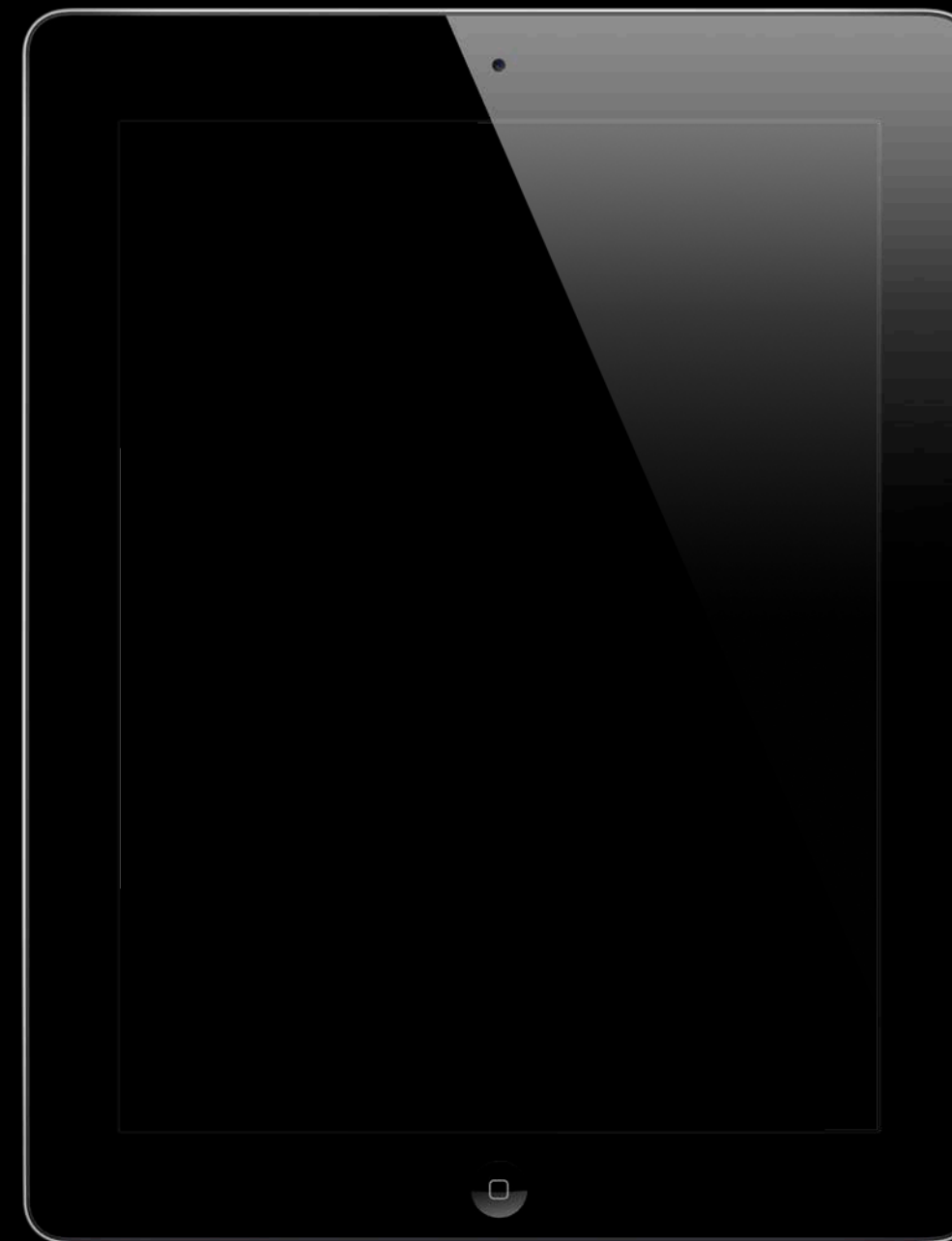
# iBeacons



# iBeacons



# iBeacons



# Jay's Donut Shop.app

Pseudocode

# Jay's Donut Shop.app

## Pseudocode

```
[locationManager startMonitoringForRegion:@"
```



```
"];
```



# Jay's Donut Shop.app

## Pseudocode

```
[locationManager startMonitoringForRegion:@"89D6E936-F61F-4227-BCD2-6E94C5A6B2E3"];
```

# Jay's Donut Shop.app

Setup

# Jay's Donut Shop.app

## Setup

```
$ uuidgen
```

```
89D6E936-F61F-4227-BCD2-6E94C5A6B2E3
```

# Jay's Donut Shop.app

## Setup

```
- (id)init
{
    if ((self = [super init]))
    {
        _jaysUUID = [[NSUUID alloc]
            initWithUUIDString:@"89D6E936-F61F-4227-BCD2-6E94C5A6B2E3"];
        _jaysId = @"Jay's Donut Shop";
        _locationManager = [[CLLocationManager alloc] init];
        _locationManager.delegate = self;
    }
    return self;
}
```

# Jay's Donut Shop.app

## Setup

```
- (id)init
{
    if ((self = [super init]))
    {
        _jaysUUID = [[NSUUID alloc]
            initWithUUIDString:@"89D6E936-F61F-4227-BCD2-6E94C5A6B2E3"];
        _jaysId = @"Jay's Donut Shop";
        _locationManager = [[CLLocationManager alloc] init];
        _locationManager.delegate = self;
    }
    return self;
}
```

# Jay's Donut Shop.app

## Setup

```
- (id)init
{
    if ((self = [super init]))
    {
        _jaysUUID = [[NSUUID alloc]
            initWithUUIDString:@"89D6E936-F61F-4227-BCD2-6E94C5A6B2E3"];
        _jaysId = @"Jay's Donut Shop";
        _locationManager = [[CLLocationManager alloc] init];
        _locationManager.delegate = self;
    }
    return self;
}
```

# Jay's Donut Shop.app

## Setup

```
- (id)init
{
    if ((self = [super init]))
    {
        _jaysUUID = [[NSUUID alloc]
            initWithUUIDString:@"89D6E936-F61F-4227-BCD2-6E94C5A6B2E3"];
        _jaysId = @"Jay's Donut Shop";
        _locationManager = [[CLLocationManager alloc] init];
        _locationManager.delegate = self;
    }
    return self;
}
```

# Jay's Donut Shop.app

## Setup

```
- (id)init
{
    if ((self = [super init]))
    {
        _jaysUUID = [[NSUUID alloc]
            initWithUUIDString:@"89D6E936-F61F-4227-BCD2-6E94C5A6B2E3"];
        _jaysId = @"Jay's Donut Shop";
        _locationManager = [[CLLocationManager alloc] init];
        _locationManager.delegate = self;
    }
    return self;
}
```



# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                     identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
```

```
{  
    CLBeaconRegion *region =  
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID  
                                           identifier:self.jaysId];  
    [_locationManager startMonitoringForRegion:region];  
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
```

```
{  
    CLBeaconRegion *region =  
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID  
                                           identifier:self.jaysId];  
    [_locationManager startMonitoringForRegion:region];  
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                     identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}

- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if ([region.identifier isEqualToString:self.jaysId])
    {
        // present notification to user
    }
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                       identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}
```

```
- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if ([region.identifier isEqualToString:self.jaysId])
    {
        // present notification to user
    }
}
```



# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                     identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}

- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if ([region.identifier isEqualToString:self.jaysId])
    {
        // present notification to user
    }
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                     identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}

- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if ([region.identifier isEqualToString:self.jaysId])
    {
        // present notification to user
    }
}
```

# Jay's Donut Shop.app

## Listening for iBeacons

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                     identifier:self.jaysId]];
    [_locationManager startMonitoringForRegion:region];
}

- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if ([region.identifier isEqualToString:self.jaysId])
    {
        // present notification to user
    }
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    region.notifyEntryStateOnDisplay = YES;
    region.notifyOnEntry = NO;
    region.notifyOnExit = YES;
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    region.notifyEntryStateOnDisplay = YES;
    region.notifyOnEntry = NO;
    region.notifyOnExit = YES;
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    region.notifyEntryStateOnDisplay = YES;
    region.notifyOnEntry = NO;
    region.notifyOnExit = YES;
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    region.notifyEntryStateOnDisplay = YES;
    region.notifyOnEntry = NO;
    region.notifyOnExit = YES;
    [_locationManager startMonitoringForRegion:region];
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)startMonitoringForStores
{
    CLBeaconRegion *region =
        [[CLBeaconRegion alloc] initWithProximityUUID:self.jaysUUID
                                         identifier:self.jaysId]];
    region.notifyEntryStateOnDisplay = YES;
    region.notifyOnEntry = NO;
    region.notifyOnExit = YES;
    [_locationManager startMonitoringForRegion:region];
}
```



# Jay's Donut Shop.app

## Notify when relevant

```
- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if (/* have not presented notification */) {
        // present notification
    }
}

- (void)locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    // clear notification
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if (/* have not presented notification */) {
        // present notification
    }
}
- (void)locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    // clear notification
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if (/* have not presented notification */) {
        // present notification
    }
}

- (void)locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    // clear notification
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if (/* have not presented notification */) {
        // present notification
    }
}

- (void)locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    // clear notification
}
```

# Jay's Donut Shop.app

## Notify when relevant

```
- (void)locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    if (/* have not presented notification */) {
        // present notification
    }
}

- (void)locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    // clear notification
}
```

# Jay's Donut Shop.app

Pickup donuts



# Jay's Donut Shop.app

Pickup donuts

- Use iPhone as receipt



# Jay's Donut Shop.app

## Pickup donuts

- Use iPhone as receipt
- Notify near register



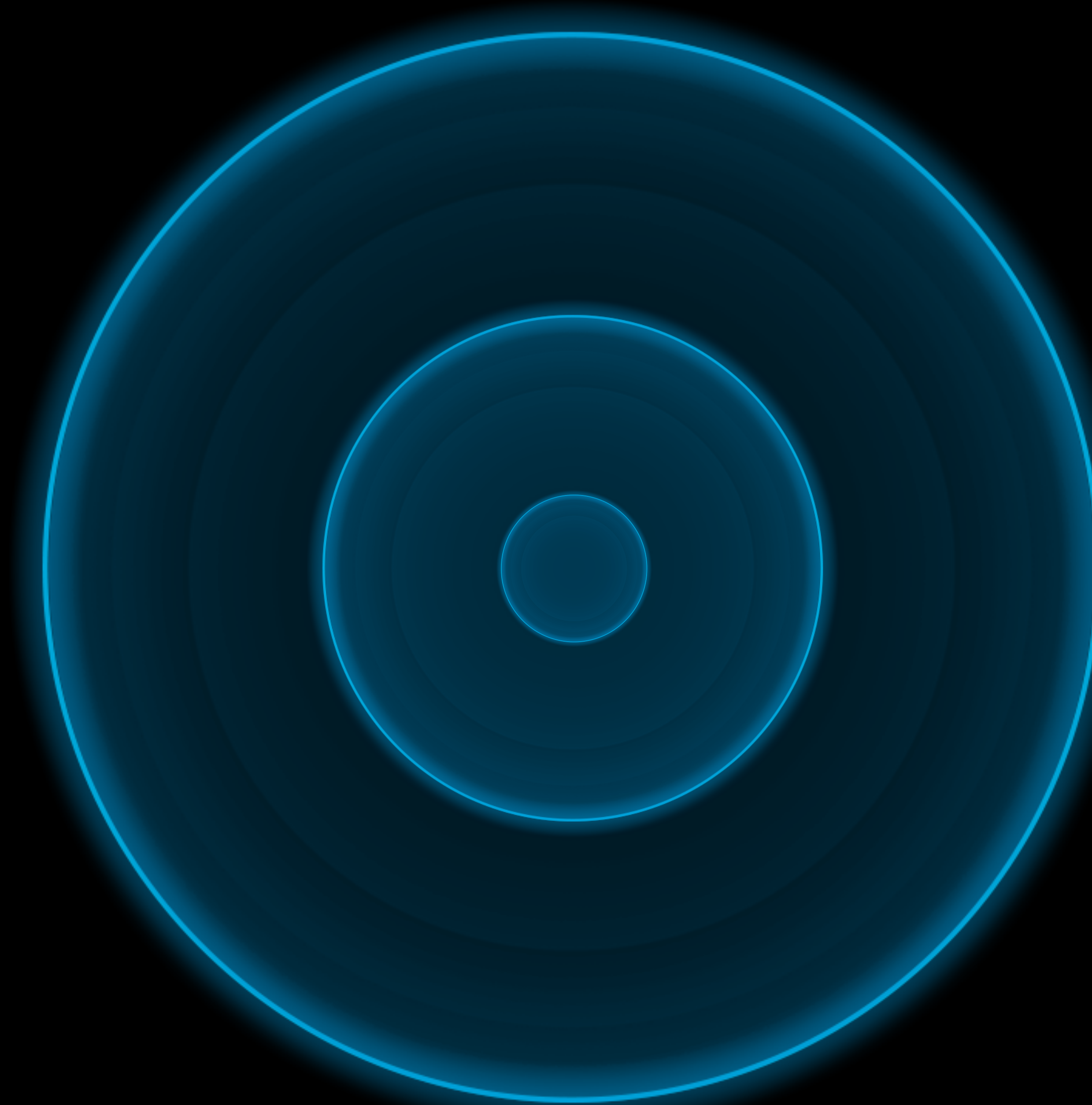


# iBeacons

## Ranging

# iBeacons

Ranging



# iBeacons

Ranging

Unknown



Immediate

Near

Far

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    [self startRangingBeaconsInRegion:region];
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    [self startRangingBeaconsInRegion:region];
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    [self startRangingBeaconsInRegion:region];
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didEnterRegion:(CLRegion *)region
{
    [self startRangingBeaconsInRegion:region];
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```



# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
        inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            [self showReceipt];
        }
    } else {
        [self hideReceipt];
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    [self.locationManager stopRangingForRegion:region];
    [self hideReceipt];
}
```



# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager  
    didExitRegion:(CLRegion *)region  
{  
    [self.locationManager stopRangingForRegion:region];  
    [self hideReceipt];  
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager  
    didExitRegion:(CLRegion *)region  
{  
    [self.locationManager stopRangingForRegion:region];  
    [self hideReceipt];  
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    [self.locationManager stopRangingForRegion:region];
    [self hideReceipt];
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didExitRegion:(CLRegion *)region
{
    [self.locationManager stopRangingForRegion:region];
    [self hideReceipt];
}
```

# iBeacons

## Ranging

```
@interface CLBeacon
@property(readonly, nonatomic) NSUUID *proximityUUID;
@property(readonly, nonatomic) CLProximity proximity;
@property(readonly, nonatomic) NSNumber *major;
@property(readonly, nonatomic) NSNumber *minor;
@end
```

```
typedef NS_ENUM(NSInteger, CLProximity) {
    CLProximityUnknown,
    CLProximityImmediate,
    CLProximityNear,
    CLProximityFar
};
```

# iBeacons

## Ranging

```
@interface CLBeacon
```

```
@property(readonly, nonatomic) NSUUID *proximityUUID;
```

```
@property(readonly, nonatomic) CLProximity proximity;
```

```
@property(readonly, nonatomic) NSNumber *major;
```

```
@property(readonly, nonatomic) NSNumber *minor;
```

```
@end
```

```
typedef NS_ENUM(NSInteger, CLProximity) {
```

```
    CLProximityUnknown,
```

```
    CLProximityImmediate,
```

```
    CLProximityNear,
```

```
    CLProximityFar
```

```
};
```

# iBeacons

## Ranging

```
@interface CLBeacon
@property(readonly, nonatomic) NSUUID *proximityUUID;
@property(readonly, nonatomic) CLProximity proximity;
@property(readonly, nonatomic) NSNumber *major;
@property(readonly, nonatomic) NSNumber *minor;
@end
```

```
typedef NS_ENUM(NSUInteger, CLProximity) {
    CLProximityUnknown,
    CLProximityImmediate,
    CLProximityNear,
    CLProximityFar
};
```

# iBeacons

## Ranging

```
@interface CLBeacon
@property(readonly, nonatomic) NSUUID *proximityUUID;
@property(readonly, nonatomic) CLProximity proximity;
@property(readonly, nonatomic) NSNumber *major;
@property(readonly, nonatomic) NSNumber *minor;
@end
```

```
typedef NS_ENUM(NSInteger, CLProximity) {
    CLProximityUnknown,
    CLProximityImmediate,
    CLProximityNear,
    CLProximityFar
};
```



# iBeacons

## Ranging

```
@interface CLBeacon
@property(readonly, nonatomic) NSUUID *proximityUUID;
@property(readonly, nonatomic) CLProximity proximity;
@property(readonly, nonatomic) NSNumber *major;
@property(readonly, nonatomic) NSNumber *minor;
@end
```

```
typedef NS_ENUM(NSInteger, CLProximity) {
    CLProximityUnknown,
    CLProximityImmediate,
    CLProximityNear,
    CLProximityFar
};
```

# iBeacons

## Ranging

```
@interface CLBeacon
@property(readonly, nonatomic) NSUUID *proximityUUID;
@property(readonly, nonatomic) CLProximity proximity;
@property(readonly, nonatomic) NSNumber *major;
@property(readonly, nonatomic) NSNumber *minor;
@end
```

```
typedef NS_ENUM(NSInteger, CLProximity) {
    CLProximityUnknown,
    CLProximityImmediate,
    CLProximityNear,
    CLProximityFar
};
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
        inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            if ([self.receipt.storeNumber isEqual:nearest.major]) {
                [self showReceipt];
            }
        }
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
        inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            if ([self.receipt.storeNumber isEqual:nearest.major]) {
                [self showReceipt];
            }
        }
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
        inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            if ([self.receipt.storeNumber isEqual:nearest.major]) {
                [self showReceipt];
            }
        }
    }
}
```

# Jay's Donut Shop.app

## Pickup donuts

```
- (void) locationManager:(CLLocationManager *)manager
    didRangeBeacons:(NSArray *)beacons
    inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0) {
        CLBeacon *nearest = [beacons objectAtIndex:0];
        if (CLProximityImmediate == nearest.proximity) {
            if ([self.receipt.storeNumber isEqual:nearest.major]) {
                [self showReceipt];
            }
        }
    }
}
```



# iBeacons

Other uses of major and minor





# iBeacons

Other uses of major and minor

Welcome to Cupertino!  
Try an Apple Fritter?





# iBeacons

Other uses of major and minor

Welcome to [Cupertino!](#)  
Try an Apple Fritter?





# iBeacons

Other uses of major and minor

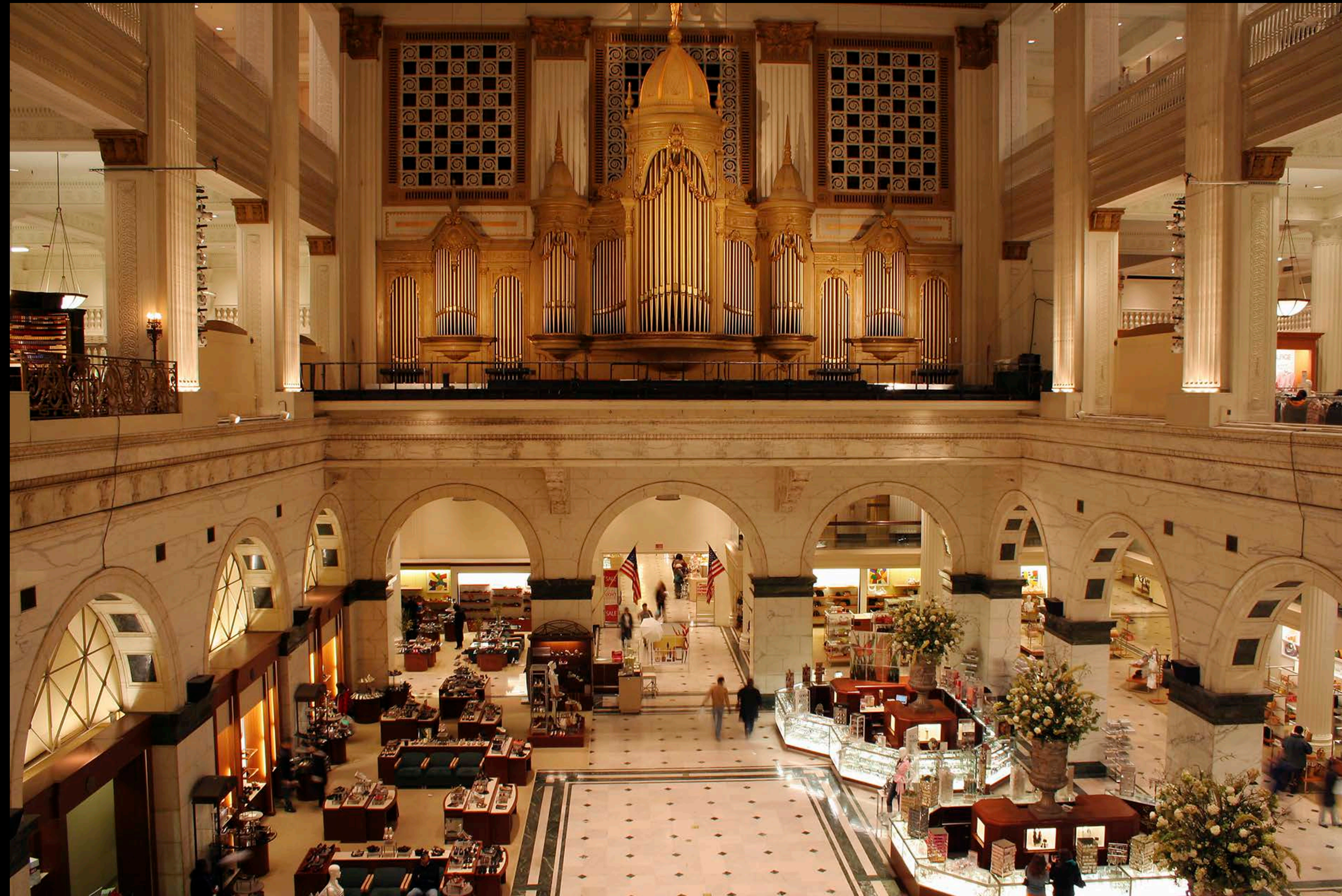
Welcome to **Cupertino!**  
Try an **Apple Fritter?**





# iBeacons

Other uses of major and minor





# iBeacons

Other uses of major and minor





# iBeacons

Other uses of major and minor





# iBeacons

Other uses of major and minor





# iBeacons

Other uses of major and minor





# iBeacons

Other uses of major and minor





# Turning iOS into an iBeacon

## Advertising

```
CLBeaconRegion *region = [[CLBeaconRegion alloc]
    initWithProximityUUID:self.jaysUUID
        major:storeNumber
        minor:donutDeal
        identifier:self.jaysId]];
NSDictionary *peripheralData = [region peripheralDataWithMeasuredPower:nil];
// CBPeripheralManager *_peripheralManager;
[self.peripheralManager startAdvertising:peripheralData];
```

# Turning iOS into an iBeacon

## Advertising

```
CLBeaconRegion *region = [[CLBeaconRegion alloc]
    initWithProximityUUID:self.jaysUUID
        major:storeNumber
        minor:donutDeal
        identifier:self.jaysId];
```

```
NSDictionary *peripheralData = [region peripheralDataWithMeasuredPower:nil];
// CBPeripheralManager *_peripheralManager;
[self.peripheralManager startAdvertising:peripheralData];
```

# Turning iOS into an iBeacon

## Advertising

```
CLBeaconRegion *region = [[CLBeaconRegion alloc]
    initWithProximityUUID:self.jaysUUID
        major:storeNumber
        minor:donutDeal
        identifier:self.jaysId]];
NSMutableDictionary *peripheralData = [region peripheralDataWithMeasuredPower:nil];
// CBPeripheralManager *_peripheralManager;
[self.peripheralManager startAdvertising:peripheralData];
```

# Turning iOS into an iBeacon

## Advertising

```
CLBeaconRegion *region = [[CLBeaconRegion alloc]
    initWithProximityUUID:self.jaysUUID
        major:storeNumber
        minor:donutDeal
        identifier:self.jaysId]];
NSDictionary *peripheralData = [region peripheralDataWithMeasuredPower:nil];
// CBPeripheralManager *_peripheralManager;
[self.peripheralManager startAdvertising:peripheralData];
```

# Turning iOS into an iBeacon

## Advertising

```
CLBeaconRegion *region = [[CLBeaconRegion alloc]
    initWithProximityUUID:self.jaysUUID
        major:storeNumber
        minor:donutDeal
        identifier:self.jaysId]];
NSDictionary *peripheralData = [region peripheralDataWithMeasuredPower:nil];
// CBPeripheralManager *_peripheralManager;
[self.peripheralManager startAdvertising:peripheralData];
```

# Turning iOS into an iBeacon

## Advertising

```
CLBeaconRegion *region = [[CLBeaconRegion alloc]
    initWithProximityUUID:self.jaysUUID
        major:storeNumber
        minor:donutDeal
        identifier:self.jaysId]];
NSDictionary *peripheralData = [region peripheralDataWithMeasuredPower:nil];
// CBPeripheralManager *_peripheralManager;
[self.peripheralManager startAdvertising:peripheralData];
```



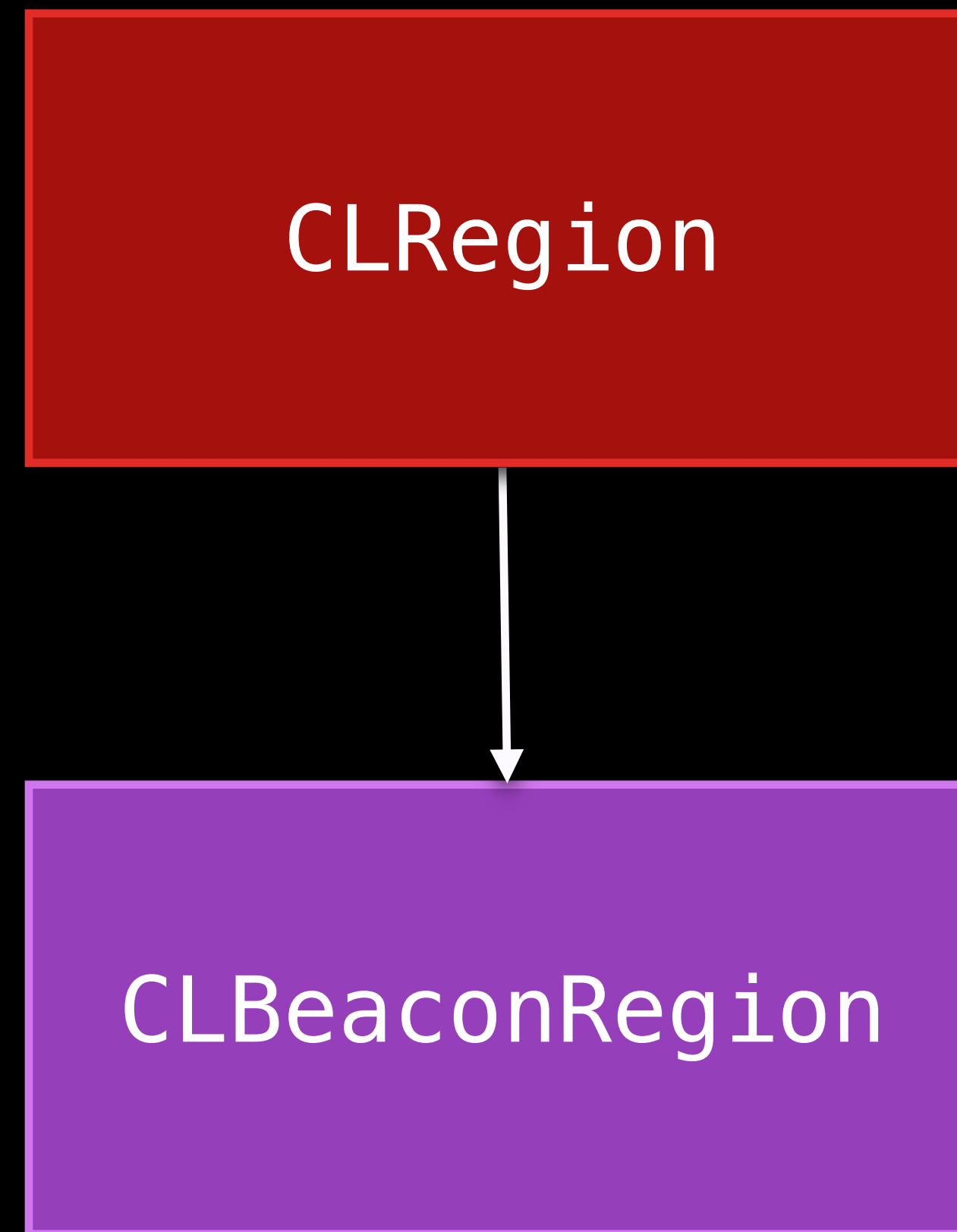
# Regions

# Regions

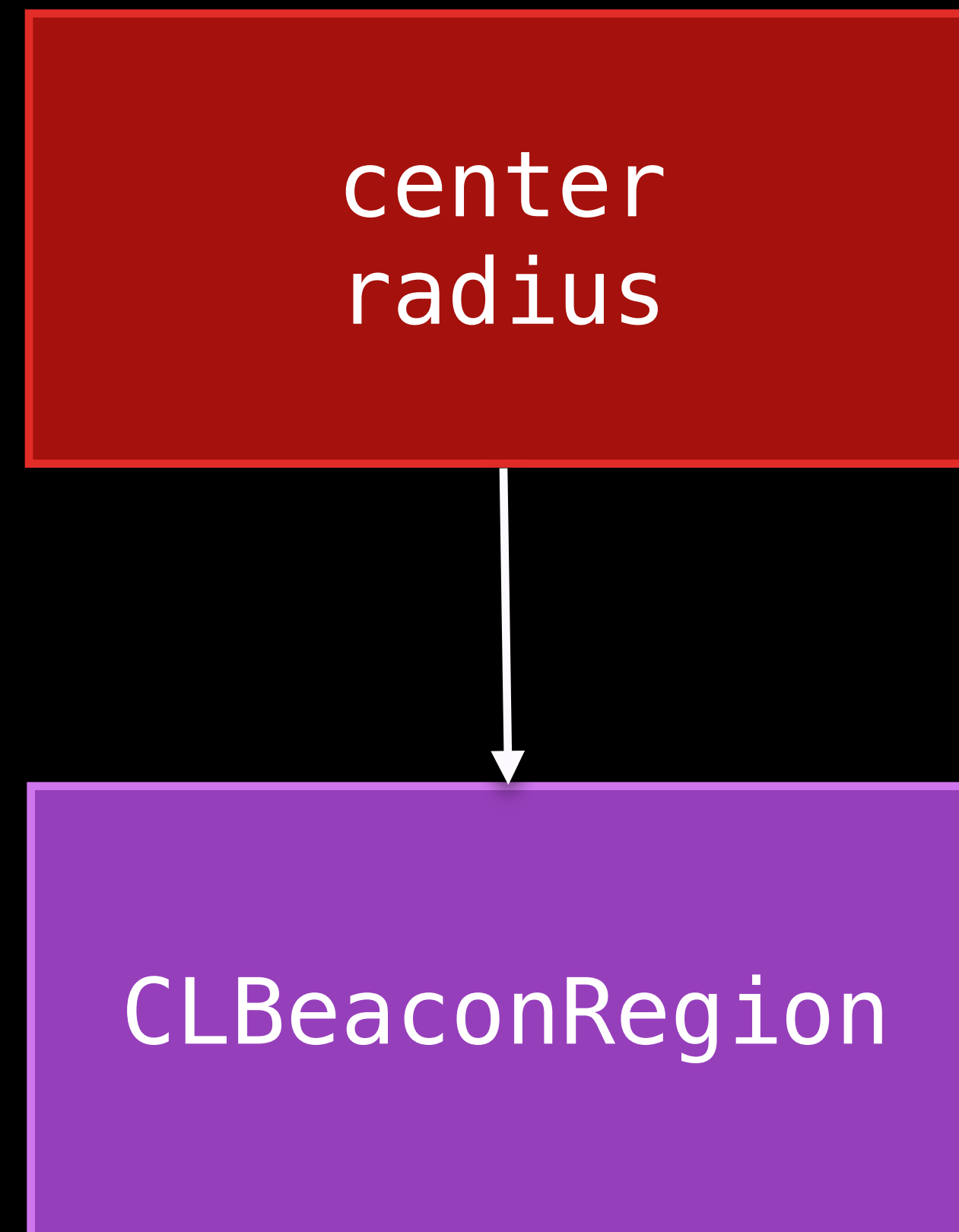


CLRegion

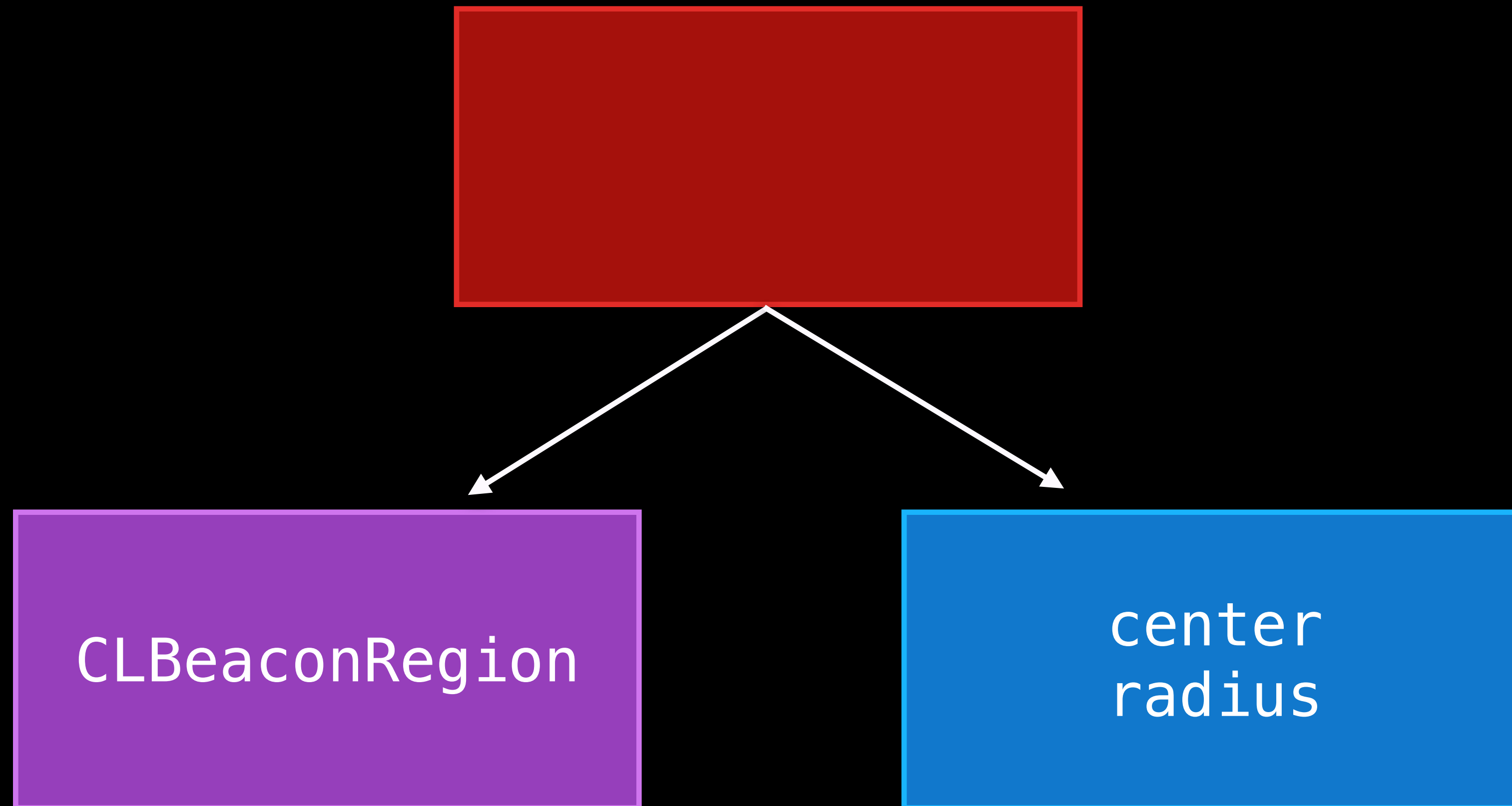
# Regions



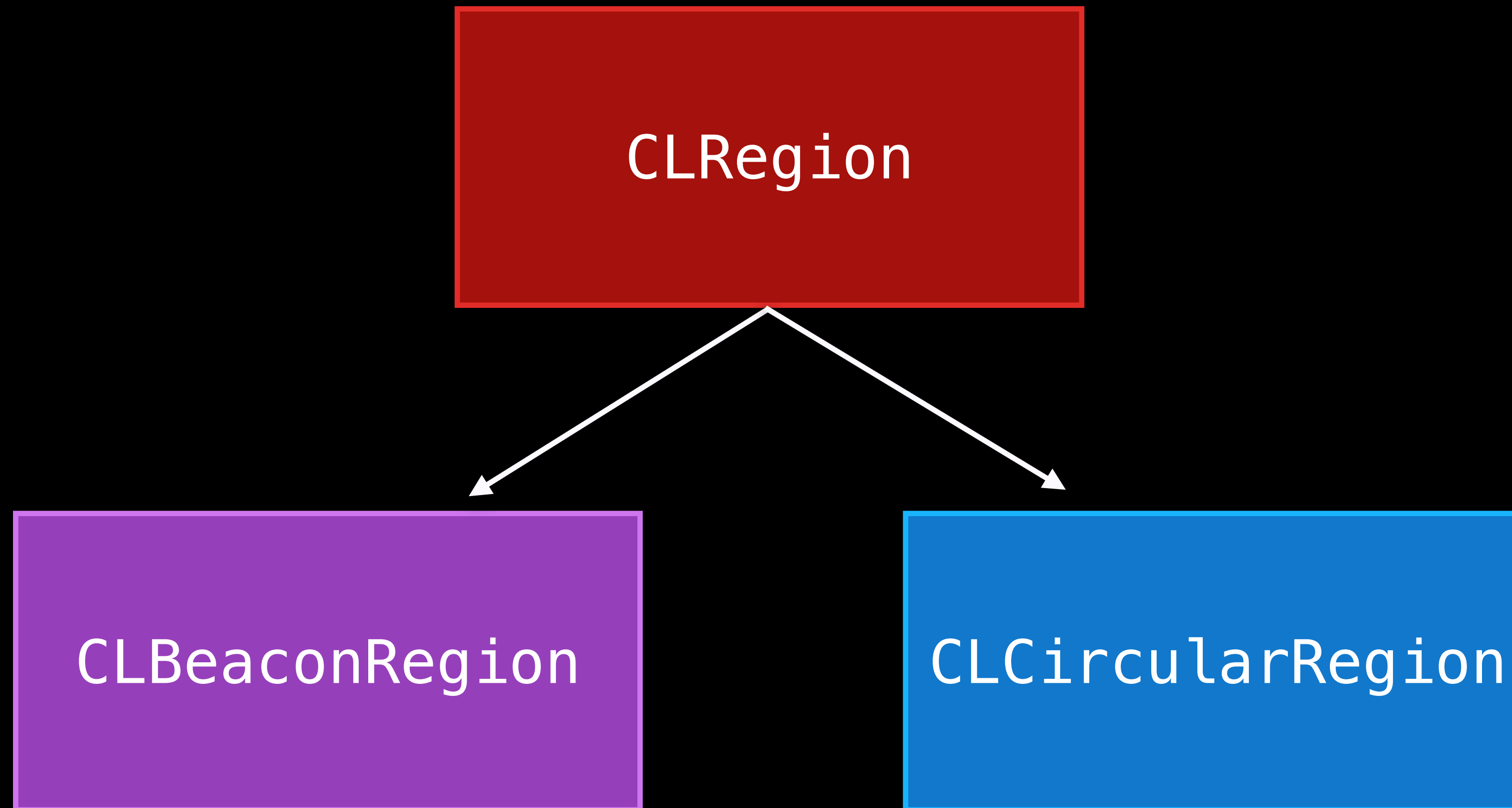
# Regions



# Regions



# Regions





# Summary

- Be aware of multitasking changes
- Defer location updates
- Use iBeacons to delight users

# More Information

## Paul Marcos

App Services Evangelist  
[pmarcos@apple.com](mailto:pmarcos@apple.com)






## Documentation

Location Awareness Programming Guide  
<http://developer.apple.com/library/ios>

## Apple Developer Forums

<http://devforums.apple.com>

# Related Sessions

What's New with Multitasking	Presidio Tuesday 2:00PM	
Core Bluetooth	Nob Hill Tuesday 2:00PM	
What's New in Map Kit	Presidio Thursday 9:00AM	
Putting Map Kit in Perspective	Pacific Heights Thursday 2:00PM	
Harnessing iOS to Create Magic in Your Apps	Presidio Friday 11:30AM	

# Labs

Core Location and iBeacons Lab

Services Lab A  
Thursday 12:45PM



 WWDC2013