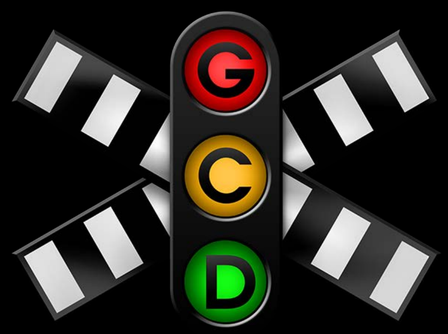
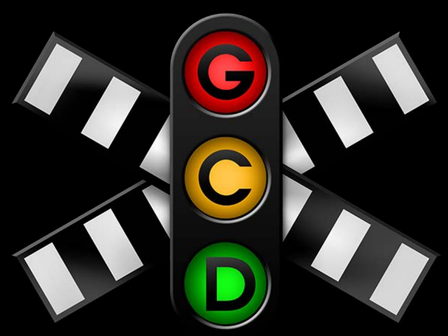
Building Responsive and Efficient Apps with GCD

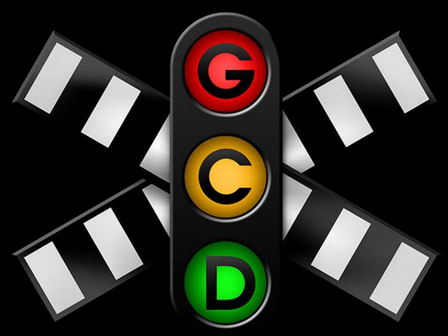
Session 718

Anthony J. Chivetta Darwin Runtime Engineer
Daniel A. Steffen Darwin Runtime Engineer


G
C
D


G
C
D

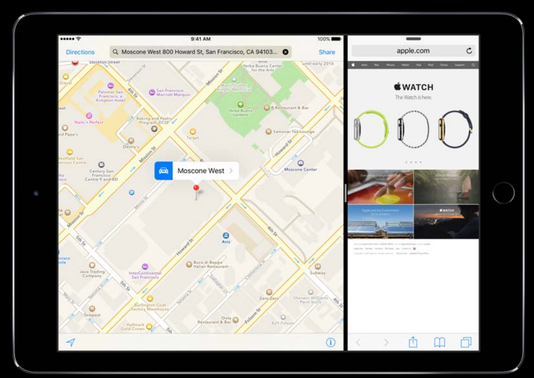

Mac OS X Snow Leopard
Mac OS X Snow Leopard
The world's most advanced operating system. Finely tuned.


G
C
D

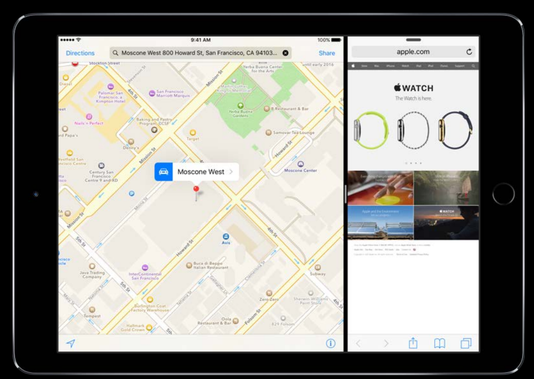

Mac OS X Snow Leopard
Mac OS X Snow Leopard
The world's most advanced operating system. Finely tuned.


MacBook Pro


Finder File Edit View Go Window Help
Thu 9:41 AM
MacBook


9:41 AM
100%
Directions
Moscone West 800 Howard St, San Francisco, CA 94103...
Share
Moscone West
apple.com
WATCH
The Watch is here.


Finder File Edit View Go Window Help
Thu 9:41 AM
MacBook


Directions
Moscone West 800 Howard St, San Francisco, CA 94103...
Share
Moscone West
apple.com
WATCH
The Watch is here.


MacBook

Quality of Service Introduction

GCD Design Patterns with QoS

Threads, Queues, and Run Loops

GCD and Crash Reports

Handling Events

MyAwesomeApp

Handling Events

MyAwesomeApp

Main Thread

Handling Events

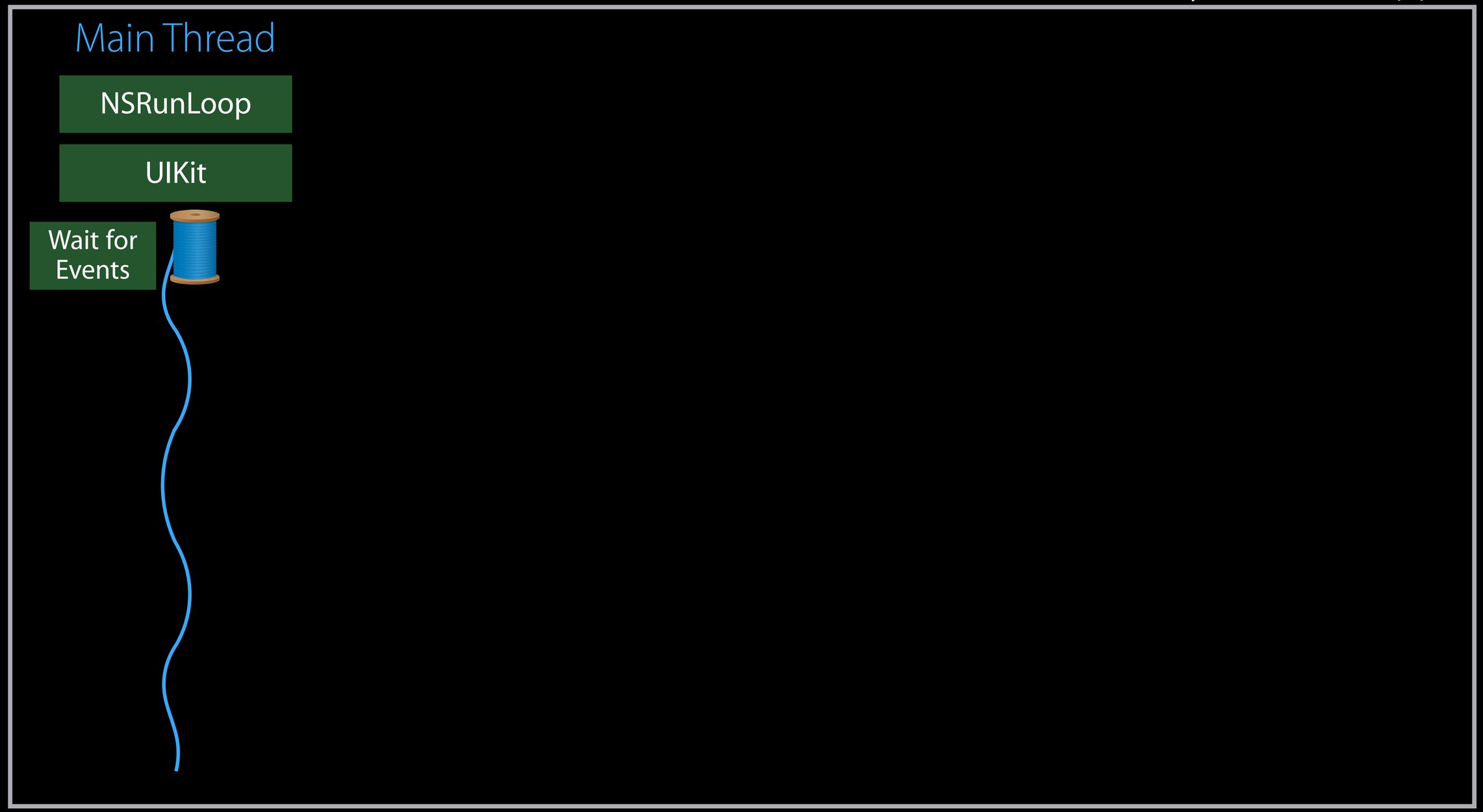
MyAwesomeApp

Main Thread

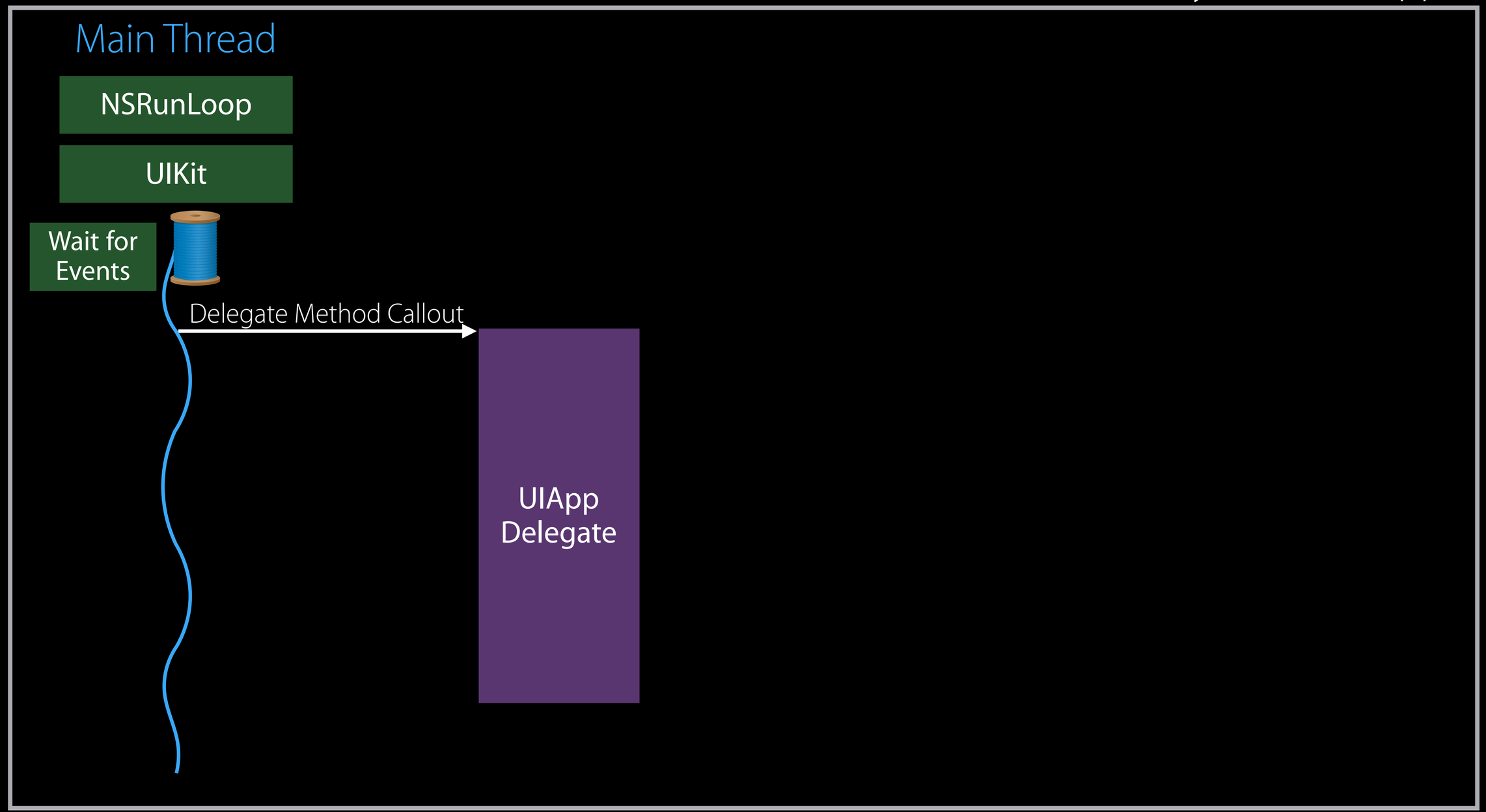
NSRunLoop

UIKit

Handling Events



Handling Events



Handling Events

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

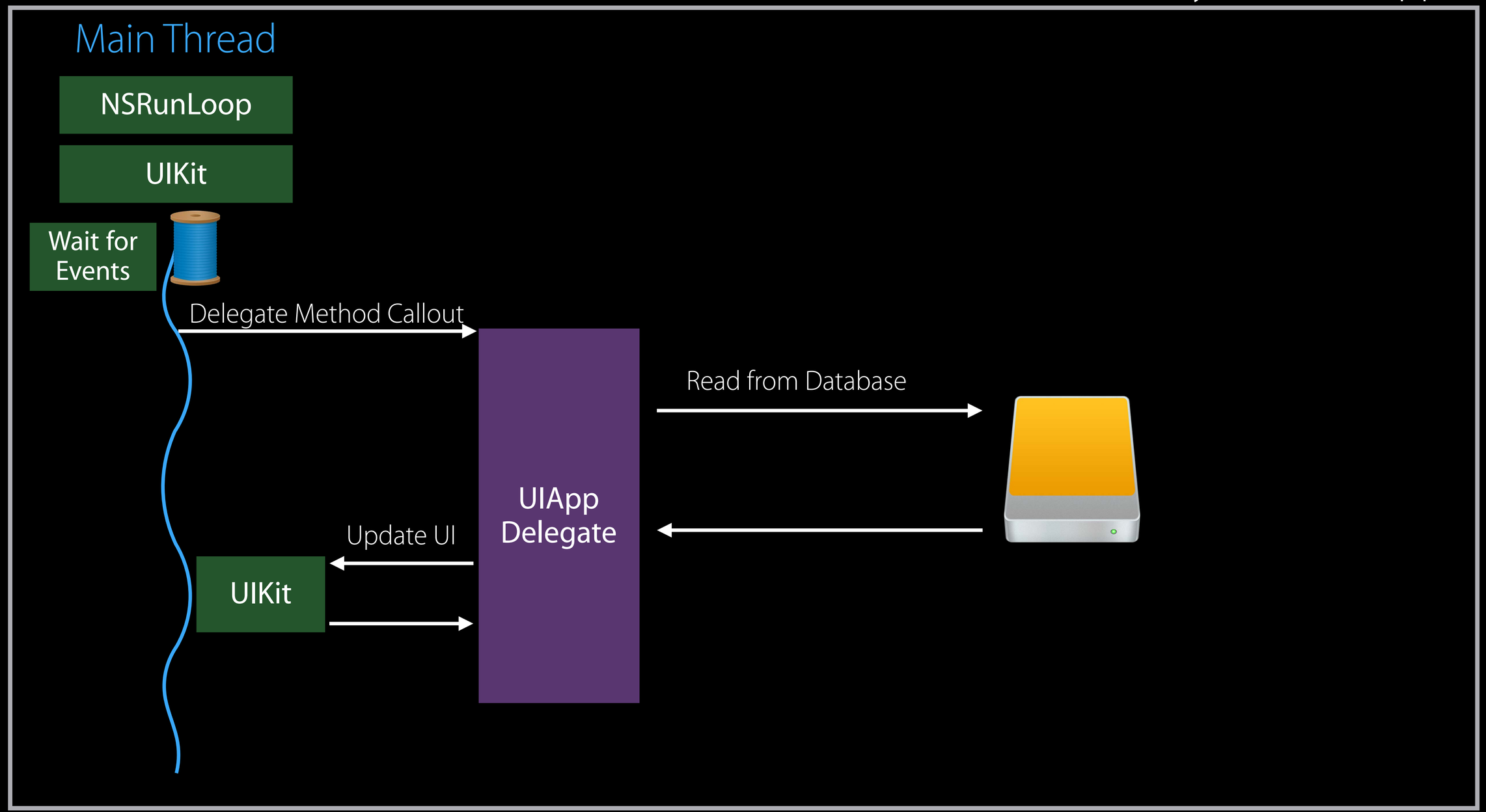
Wait for Events

Delegate Method Callout

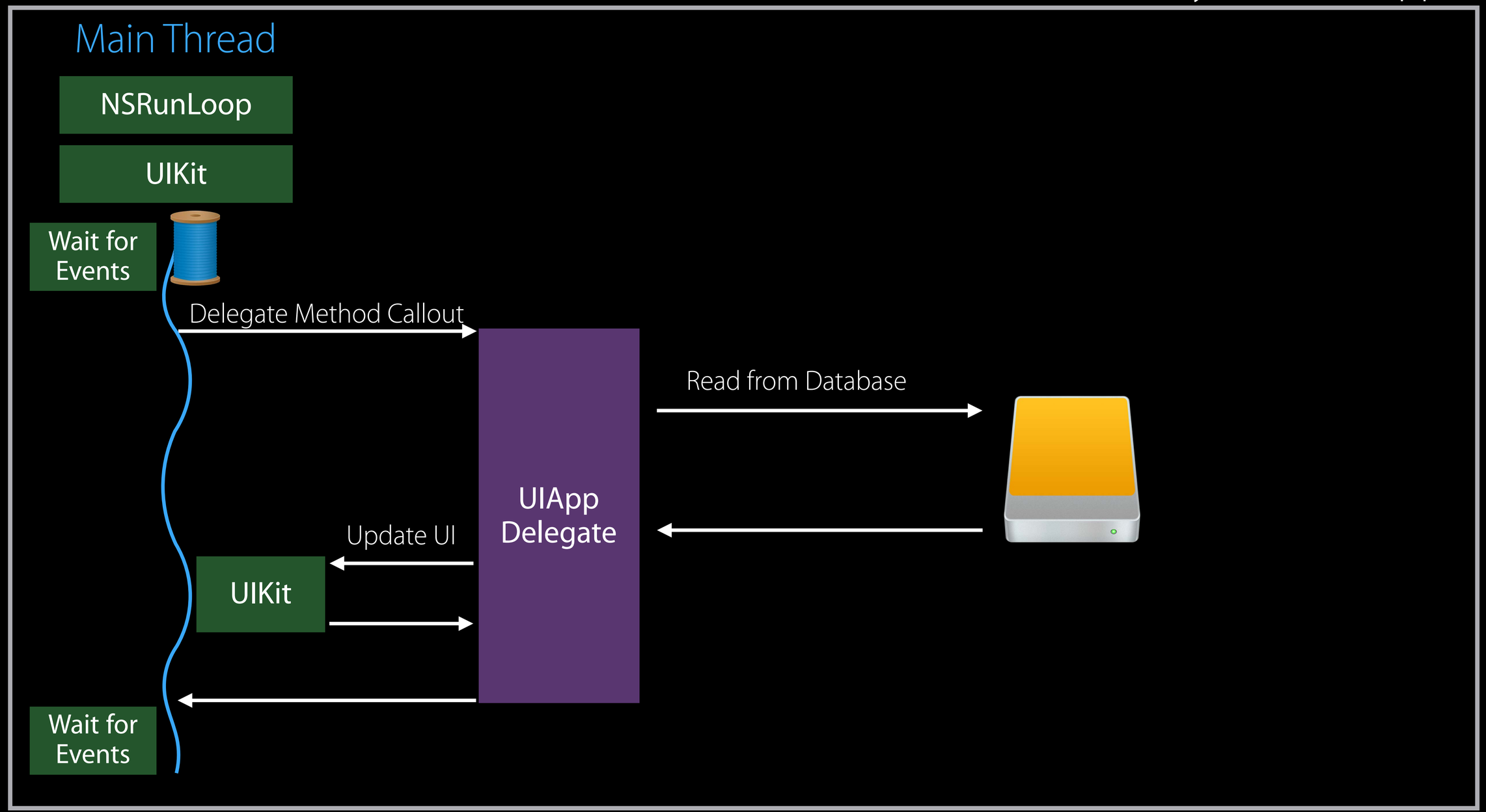
UIApp Delegate

Read from Database

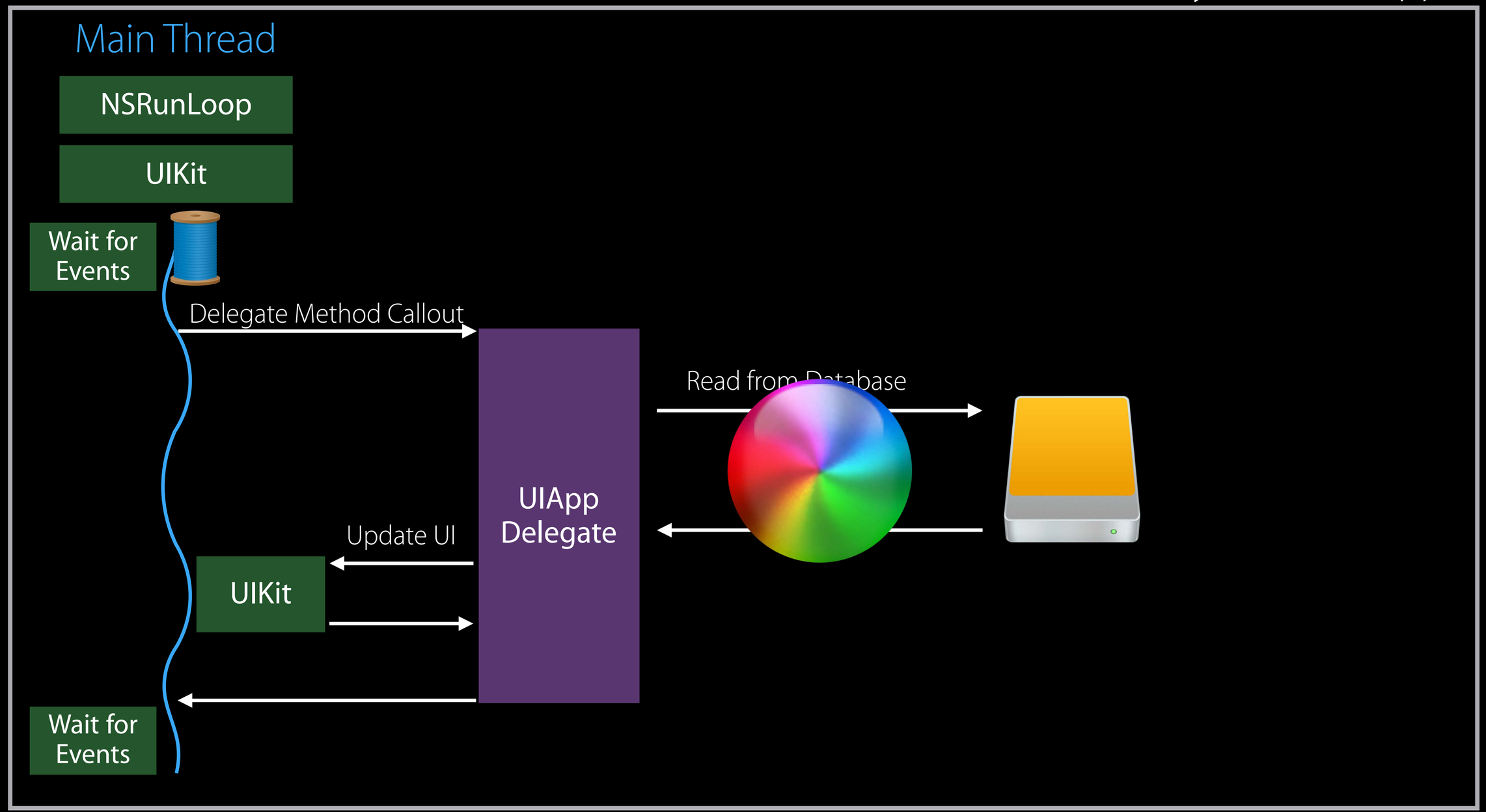
Handling Events



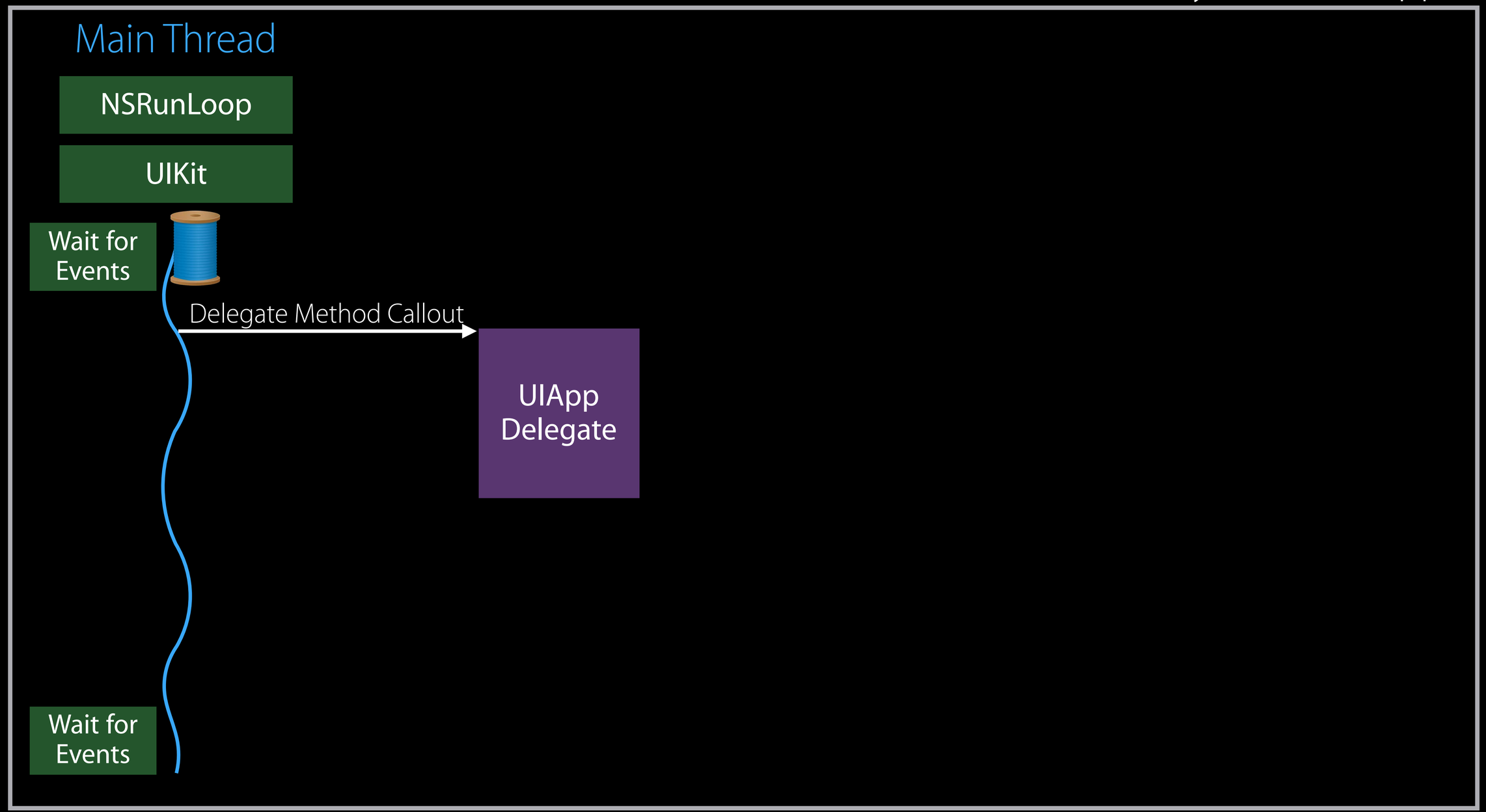
Handling Events



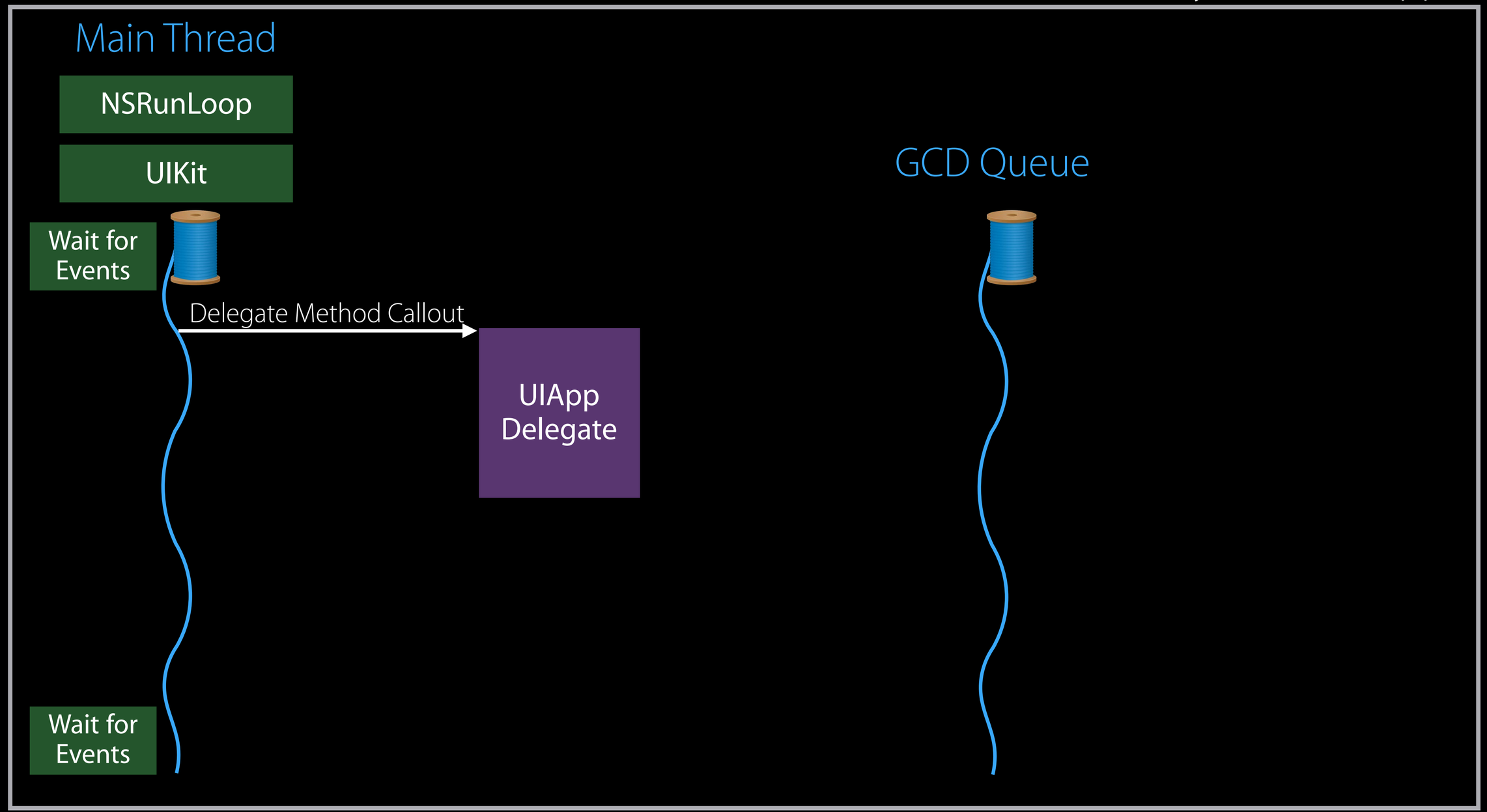
Handling Events



Handling Events Asynchronously



Handling Events Asynchronously



Handling Events Asynchronously

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

Delegate Method Callout

UIApp Delegate

dispatch_async()

GCD Queue

^{...}

Wait for Events

Handling Events Asynchronously

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

Delegate Method Callout

UIApp Delegate

dispatch_async()

GCD Queue

^{...}

dispatch_async()

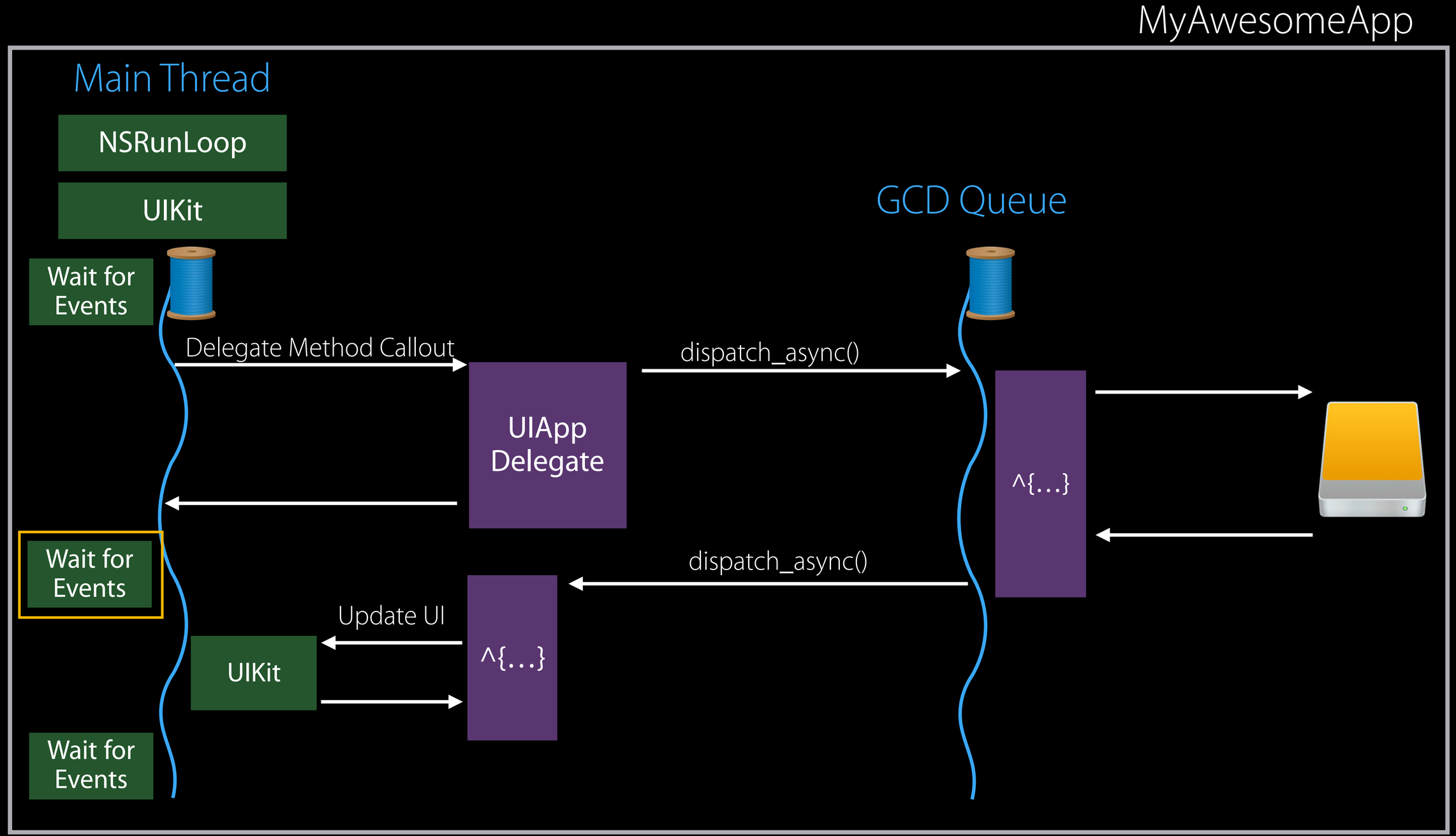
Update UI

UIKit

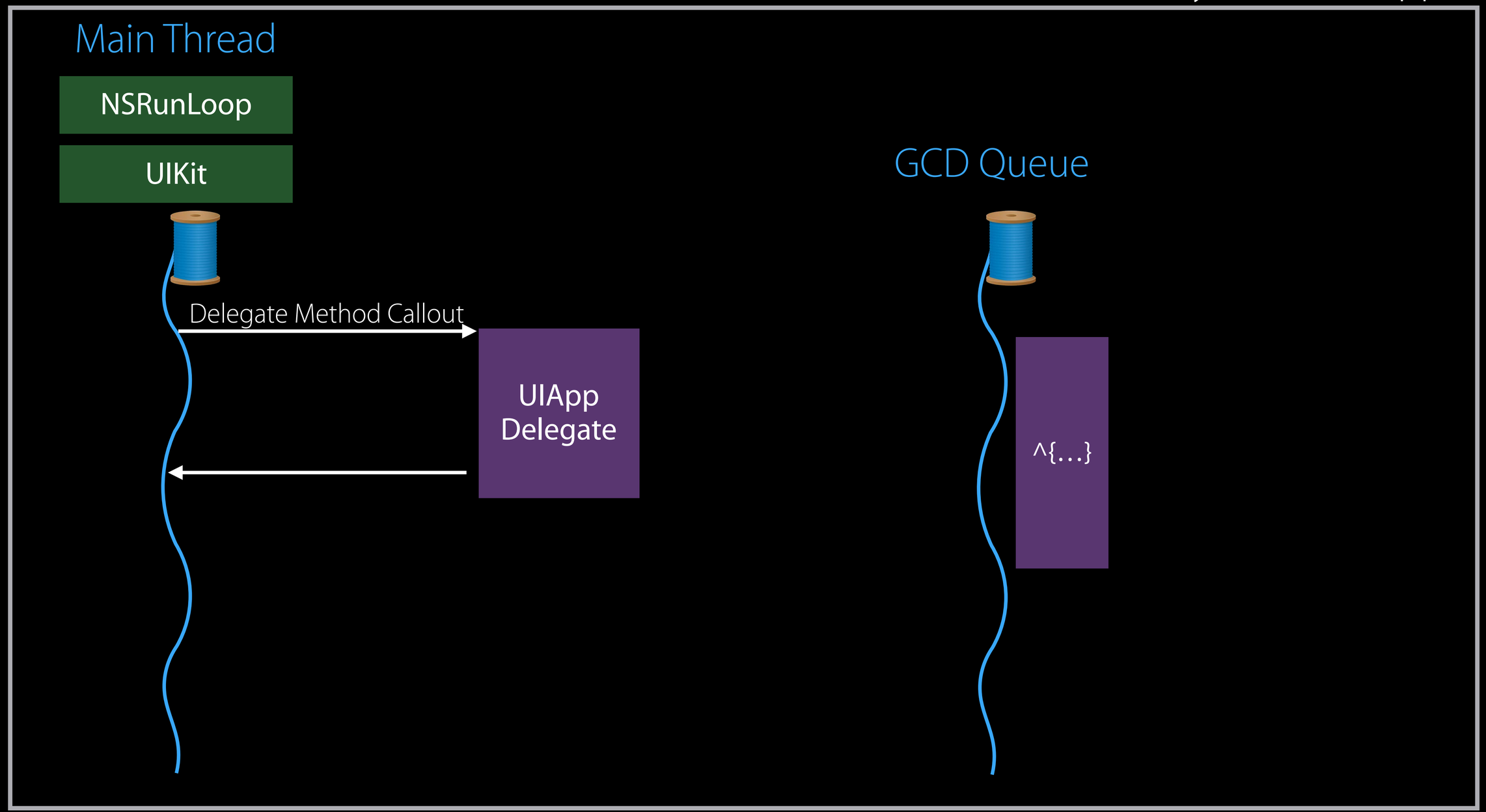
^{...}

Wait for Events

Handling Events Asynchronously



Competing Threads



Competing Threads

MyAwesomeApp

Main Thread

NSRunLoop

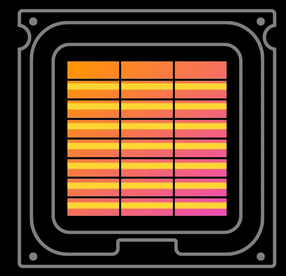
UIKit

Delegate Method Callout

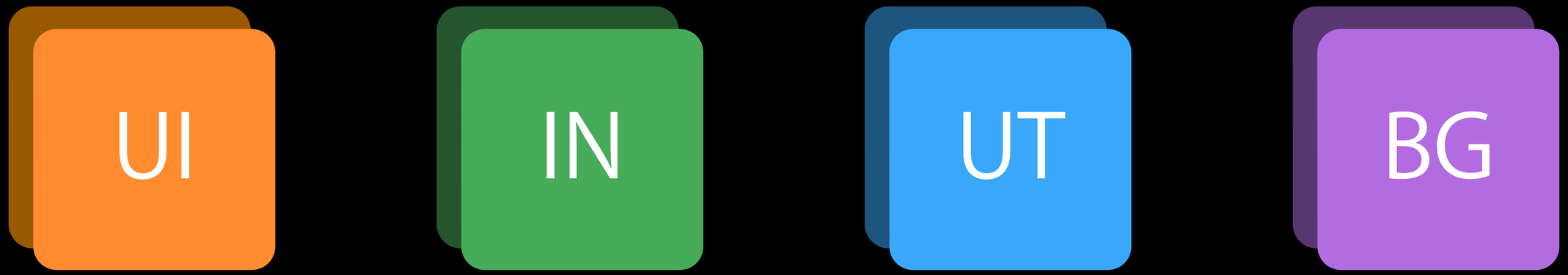
UIApp Delegate

GCD Queue

^{...}



Quality of Service Classes



User Interactive

User Initiated

Utility

Background

Quality of Service Classes

Complex resource controls

CPU scheduling priority

I/O priority

Timer coalescing

CPU throughput vs. efficiency

More…

Quality of Service Classes

Complex resource controls

CPU scheduling priority

I/O priority

Timer coalescing

CPU throughput vs. efficiency

More…

Configuration values tuned
for each platform/device

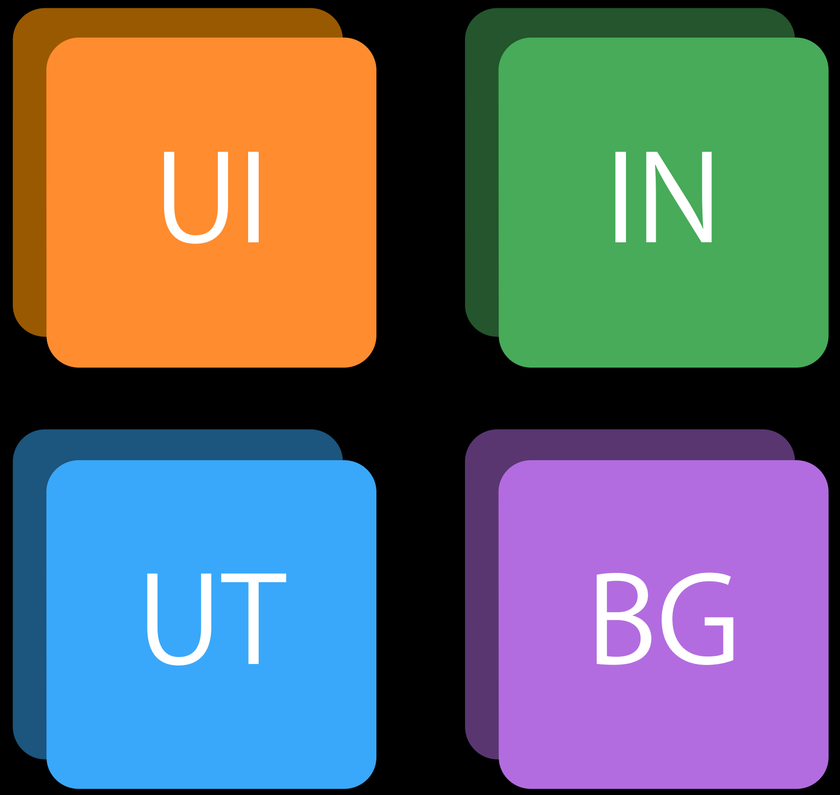
Quality of Service Classes

Single abstract parameter

Communicate developer intent

Explicit classification of work

- Move away from dictating specific configuration values



Quality of Service Classes

User Interactive	Main thread, animations
User Initiated	Immediate results
Utility	Long-running tasks
Background	Not user visible

Quality of Service Classes

User Interactive	Is this work actively involved in updating the UI?
User Initiated	Immediate results
Utility	Long-running tasks
Background	Not user visible

Quality of Service Classes

User Interactive	Is this work actively involved in updating the UI?
User Initiated	Is this work required to continue user interaction?
Utility	Long-running tasks
Background	Not user visible

Quality of Service Classes

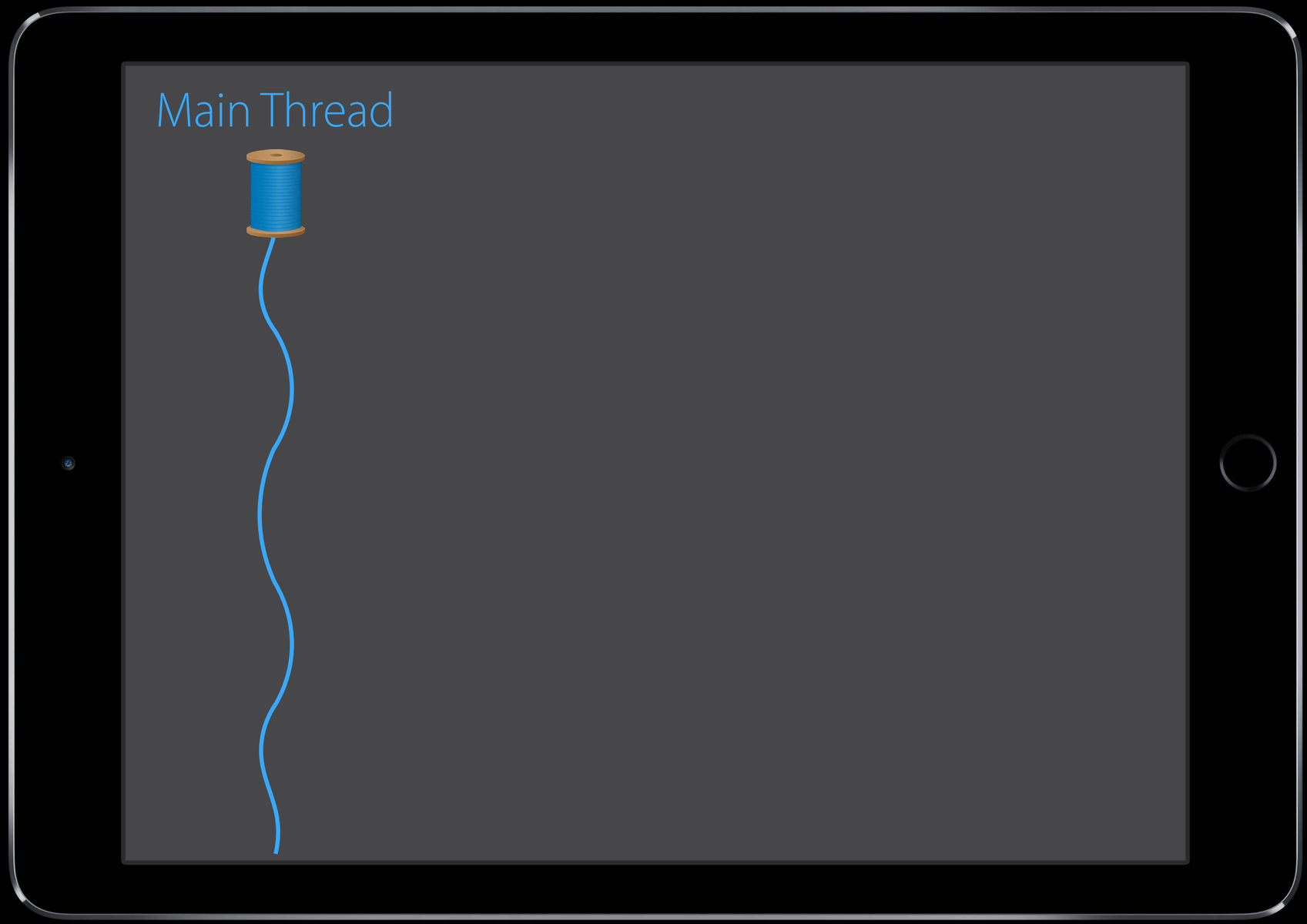
User Interactive	Is this work actively involved in updating the UI?
User Initiated	Is this work required to continue user interaction?
Utility	Is the user aware of the progress of this work?
Background	Not user visible

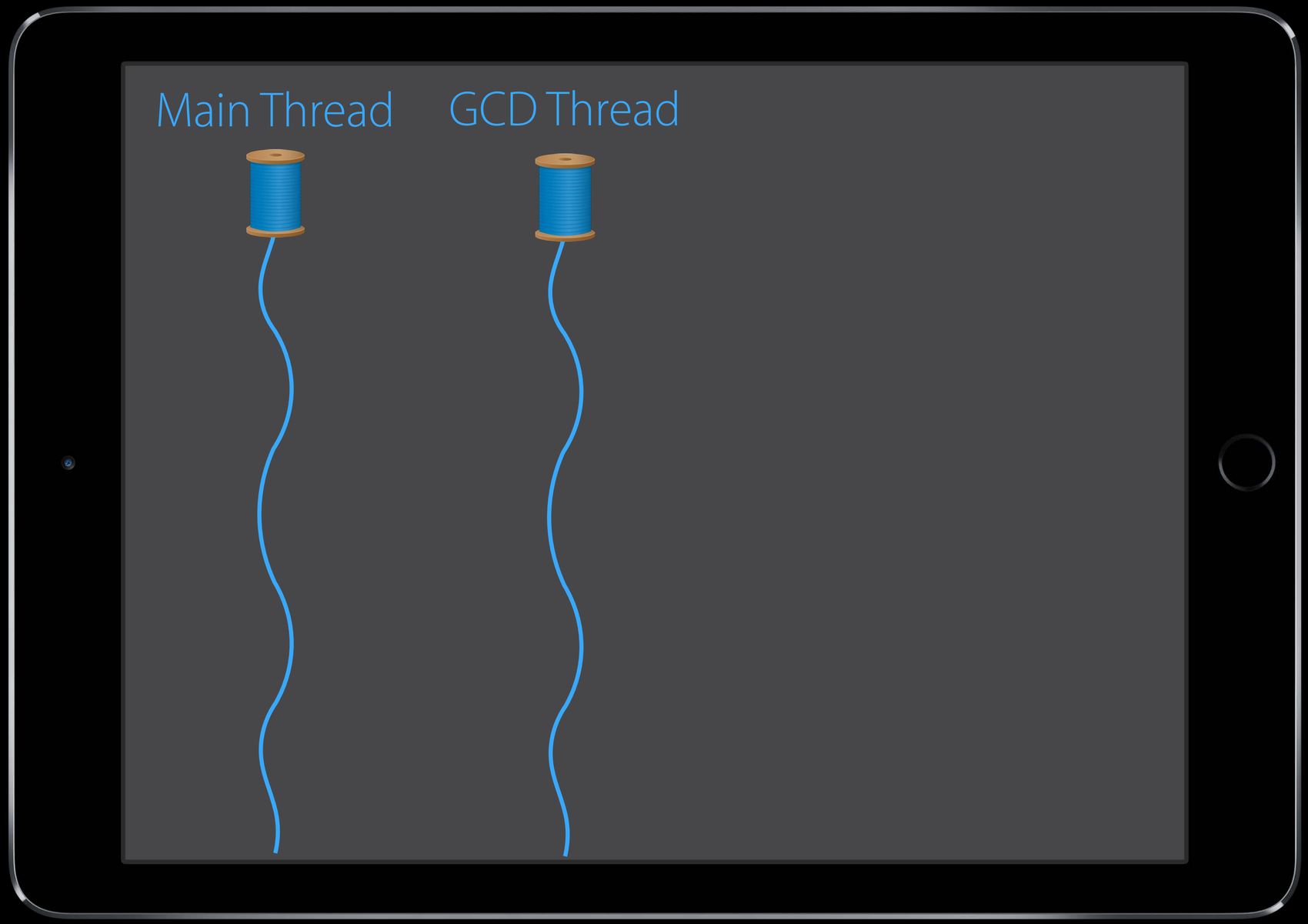
Quality of Service Classes

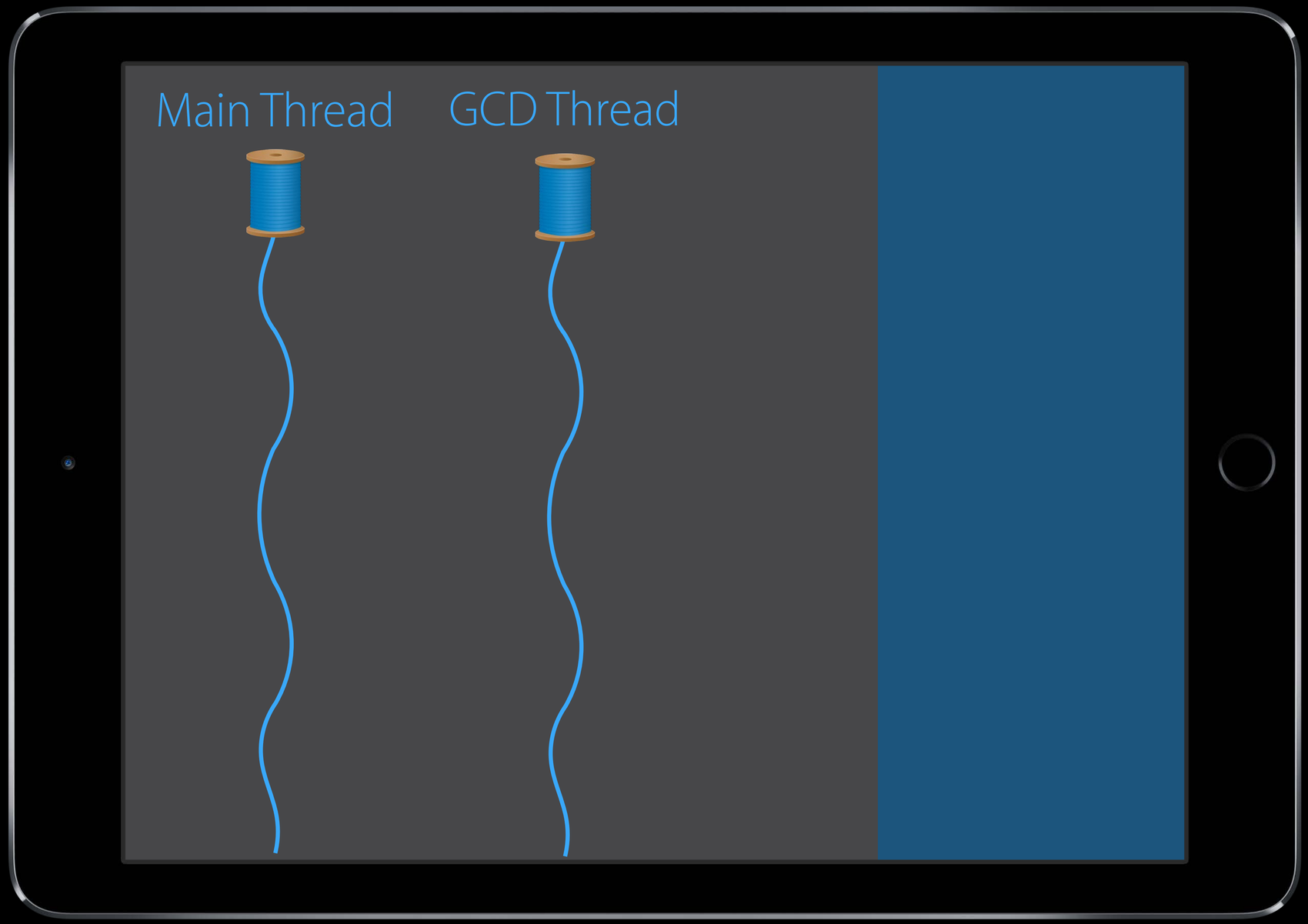
User Interactive	Is this work actively involved in updating the UI?
User Initiated	Is this work required to continue user interaction?
Utility	Is the user aware of the progress of this work?
Background	Is the user unaware of this work?

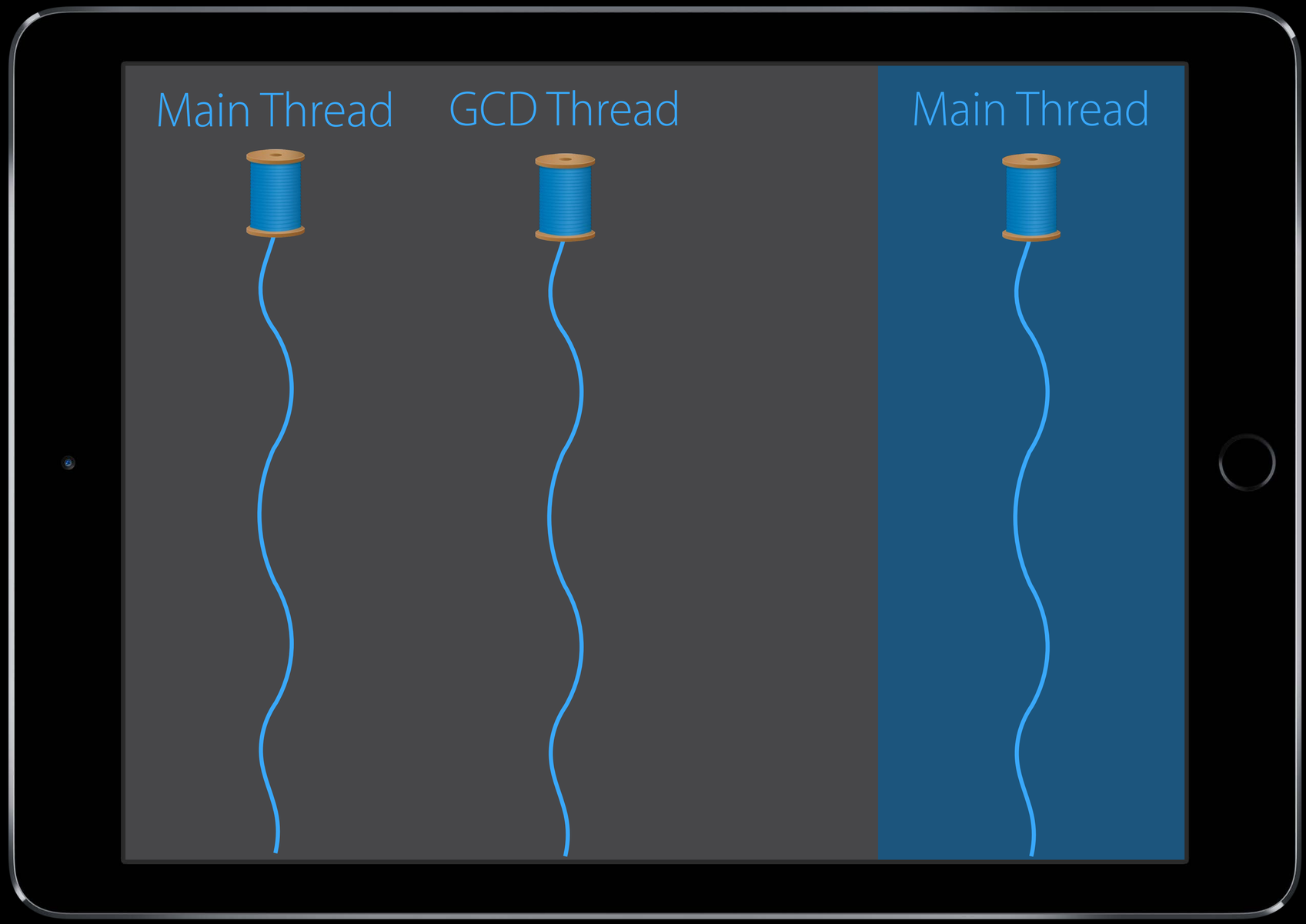

Finder File Edit View Go Window Help
Thu 9:41 AM
UI
IN
UT
BG
MacBook

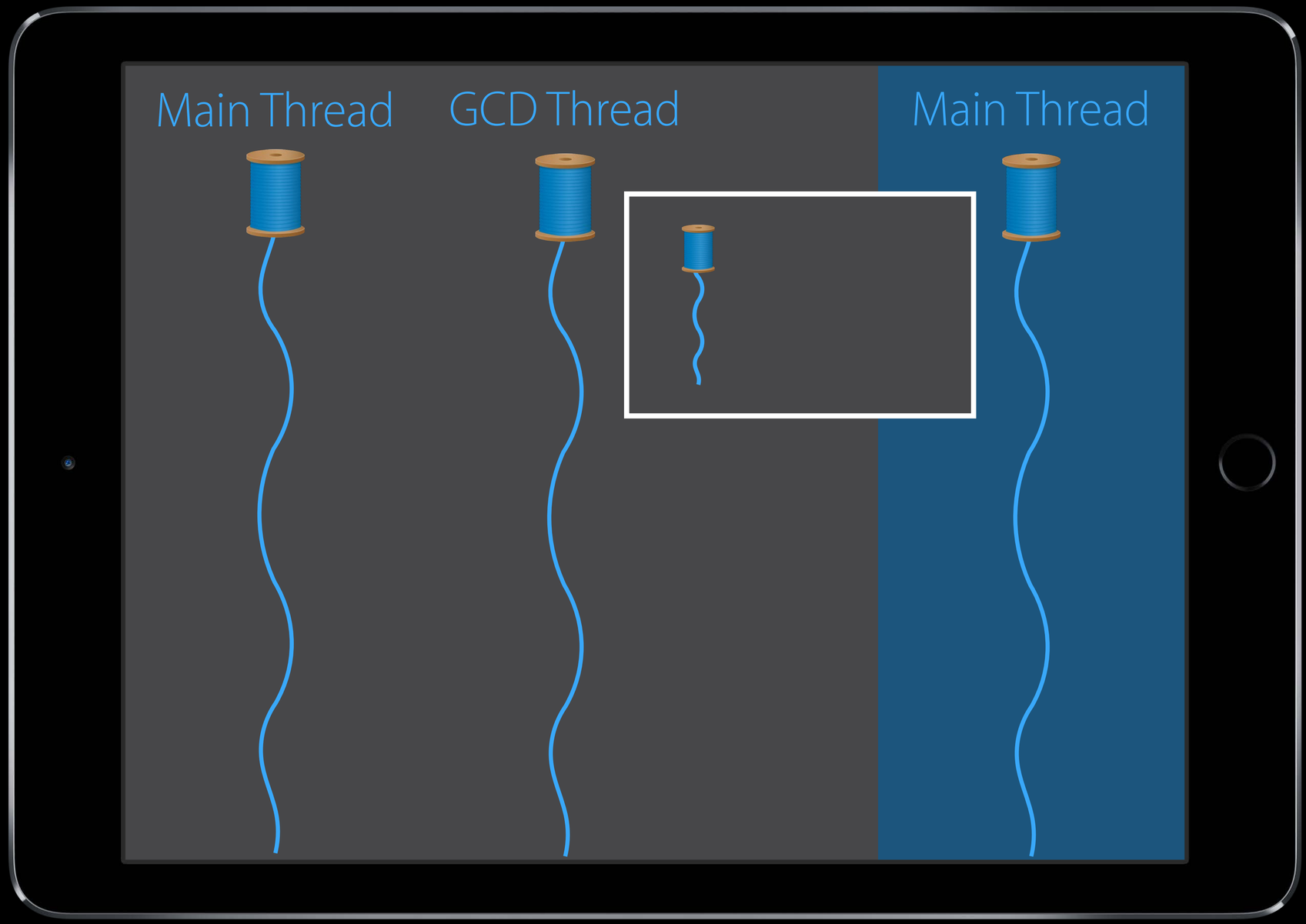

Finder File Edit View Go Window Help
Thu 9:41 AM
UI
IN
UT
BG
MacBook

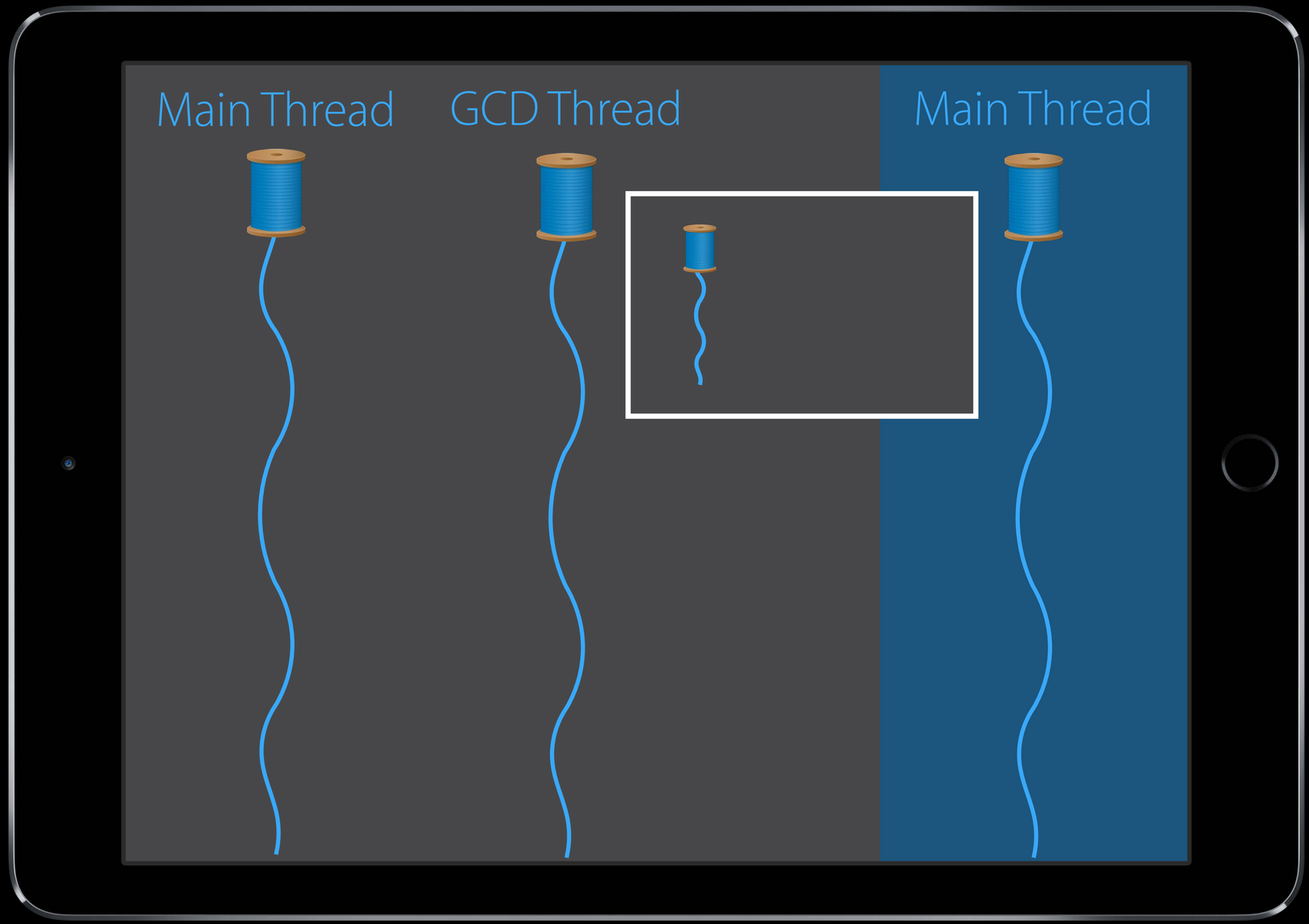

Main Thread

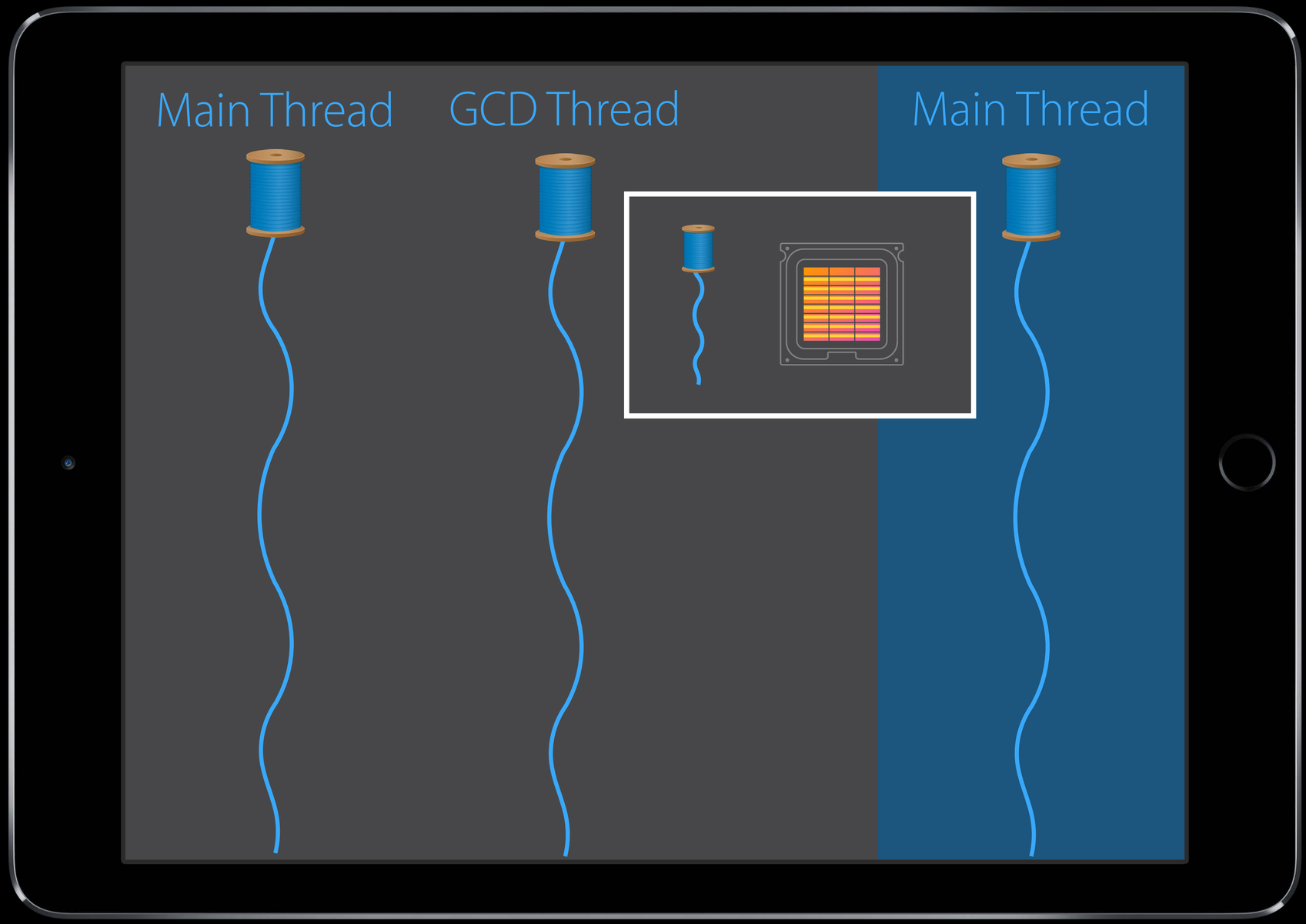

Main Thread
GCD Thread


Main Thread
GCD Thread


Main Thread
GCD Thread
Main Thread


Main Thread
GCD Thread
Main Thread


Main Thread
GCD Thread
Main Thread


Main Thread
GCD Thread
Main Thread

GCD Design Patterns with QoS

Daniel A. Steffen Darwin Runtime Engineer

GCD and QoS

Fundamentals

QoS can be specified on Blocks and on queues

dispatch_async() automatically propagates QoS

Some priority inversions are resolved automatically

Power, Performance, and Diagnostics:
What's New in GCD and XPC

WWDC14

Asynchronous Work

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

Delegate Method Callout

UIApp Delegate

dispatch_async()

GCD Queue

^{…}

Wait for Events

dispatch_async()

Update UI

UIKit

^{…}

Wait for Events

Asynchronous Work

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UI

Wait for Events

Delegate Method Callout

UIApp Delegate

dispatch_async()

GCD Queue

^{...}

Wait for Events

dispatch_async()

Update UI

UIKit

^{...}

Wait for Events

Asynchronous Work

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

Delegate Method Callout

UIApp Delegate

dispatch_async()

GCD Queue

^{...}

Wait for Events

dispatch_async()

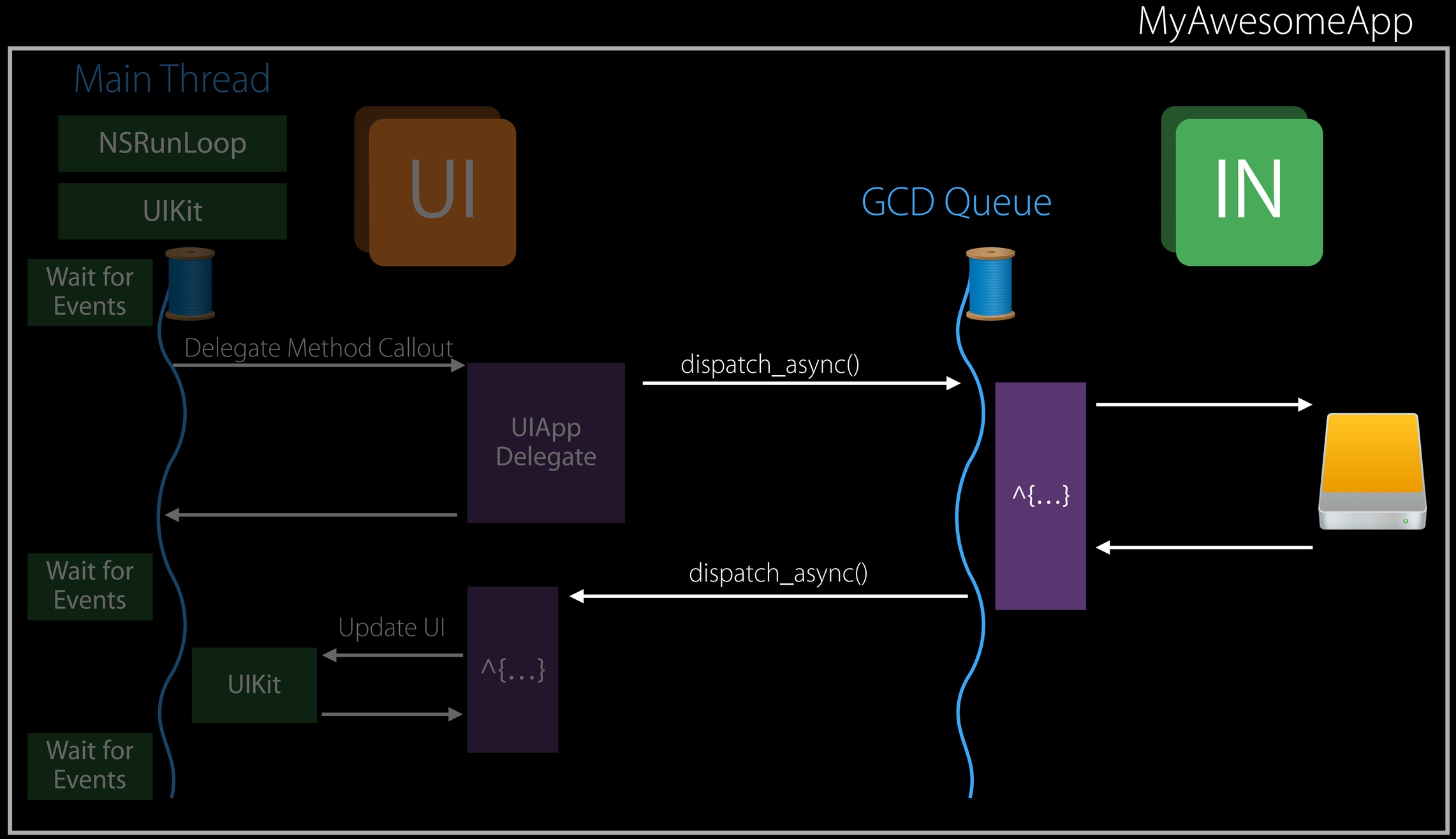
Update UI

UIKit

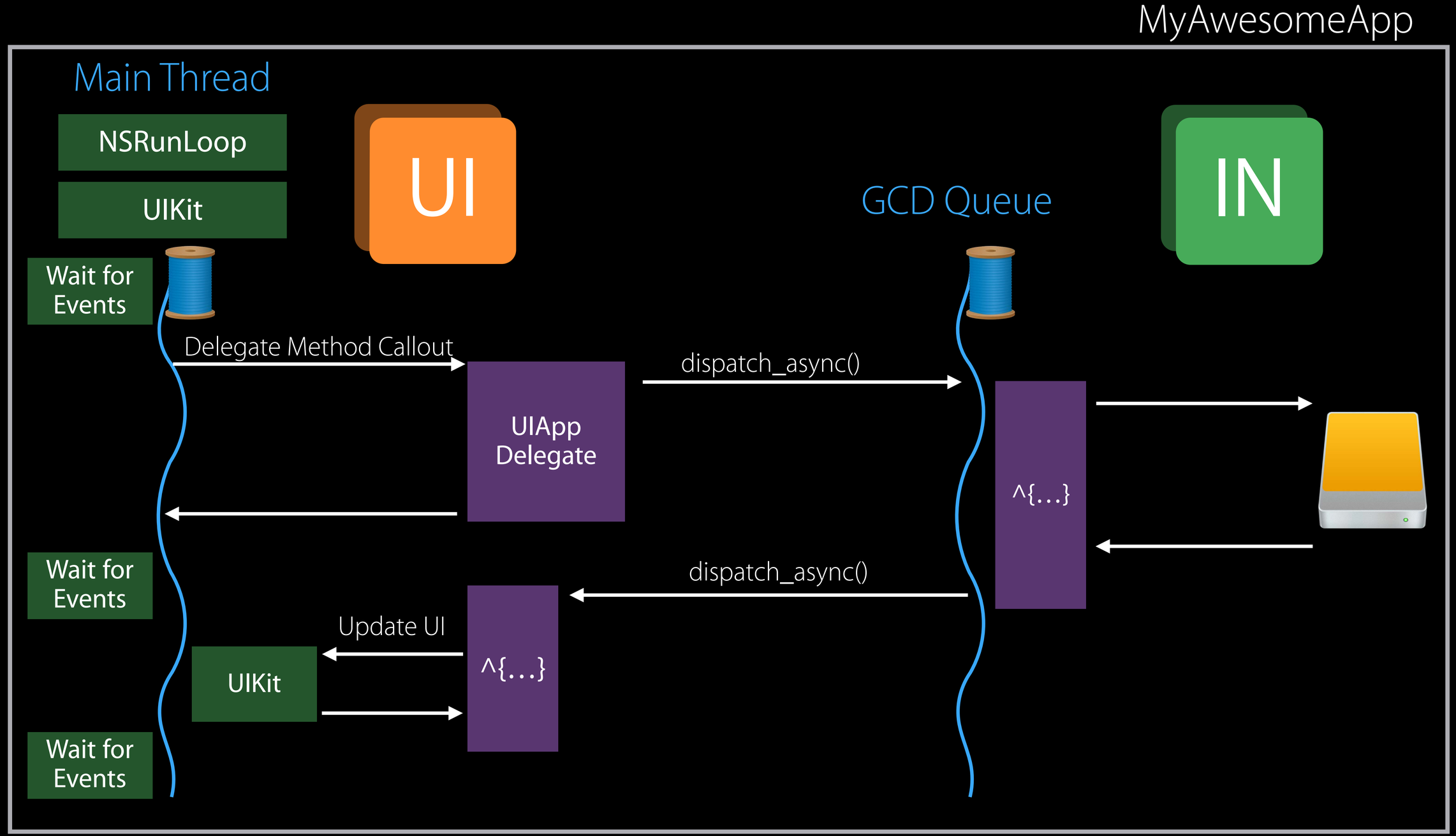
^{...}

Wait for Events

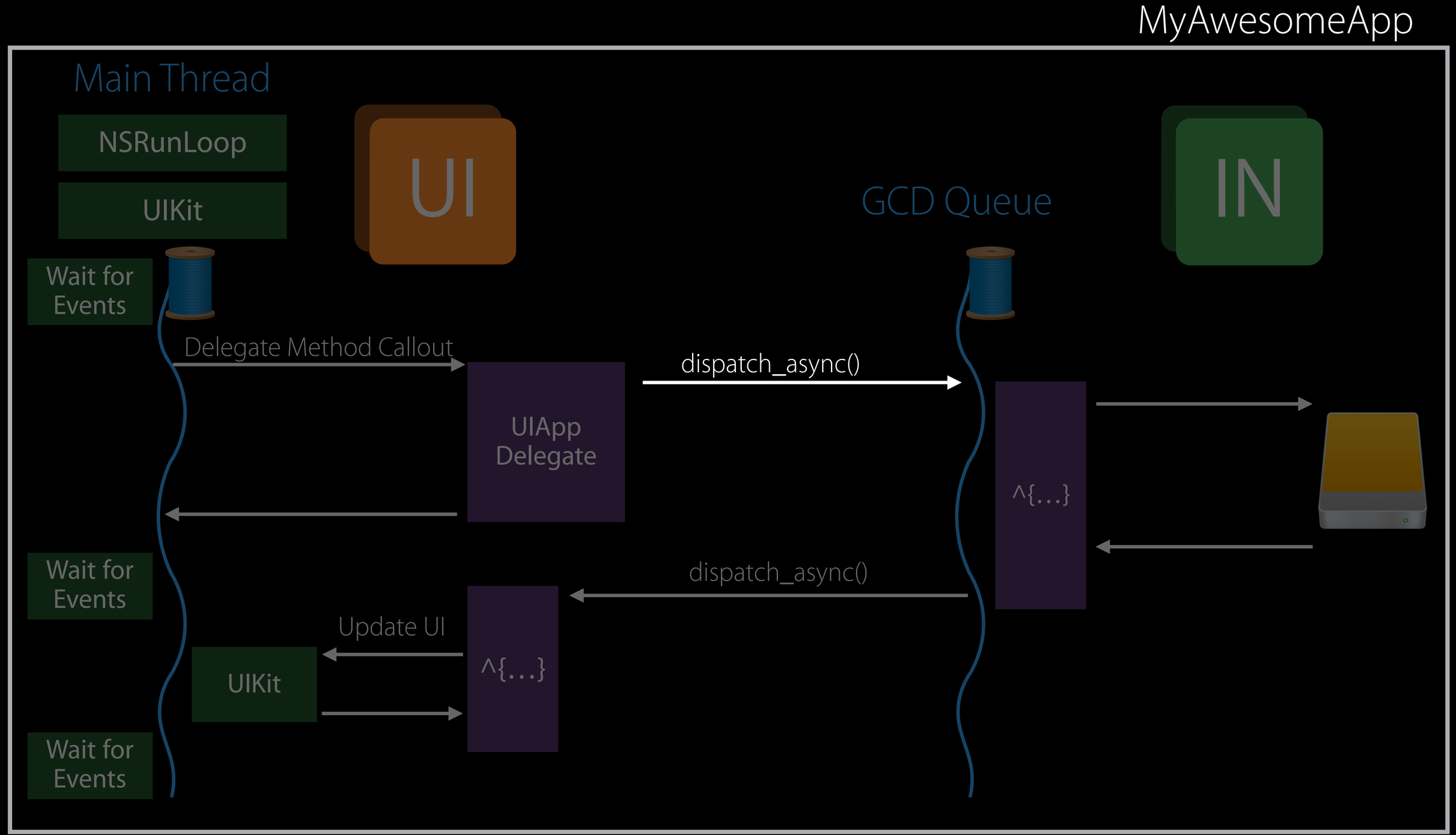
Asynchronous Work



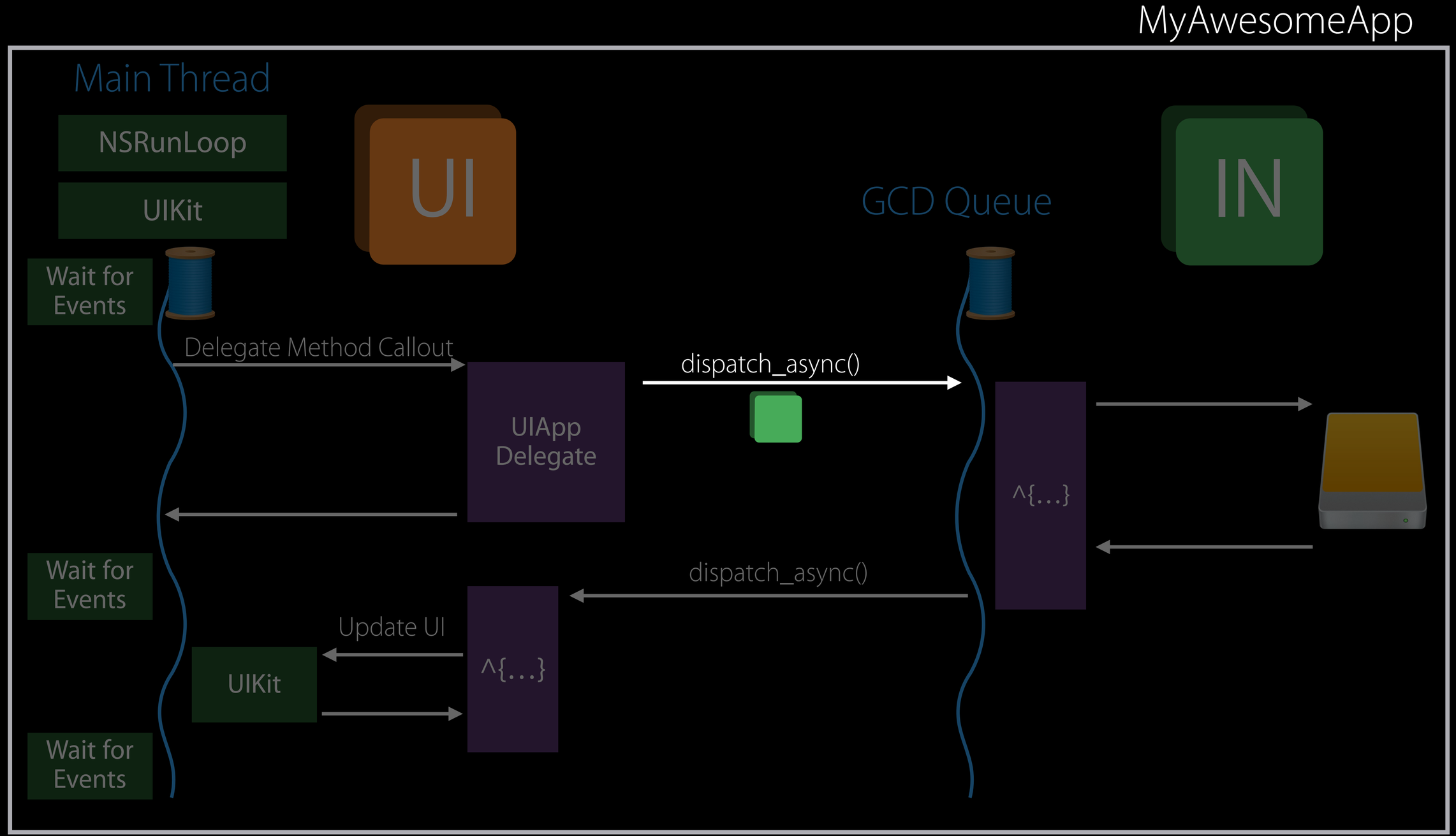
Asynchronous Work



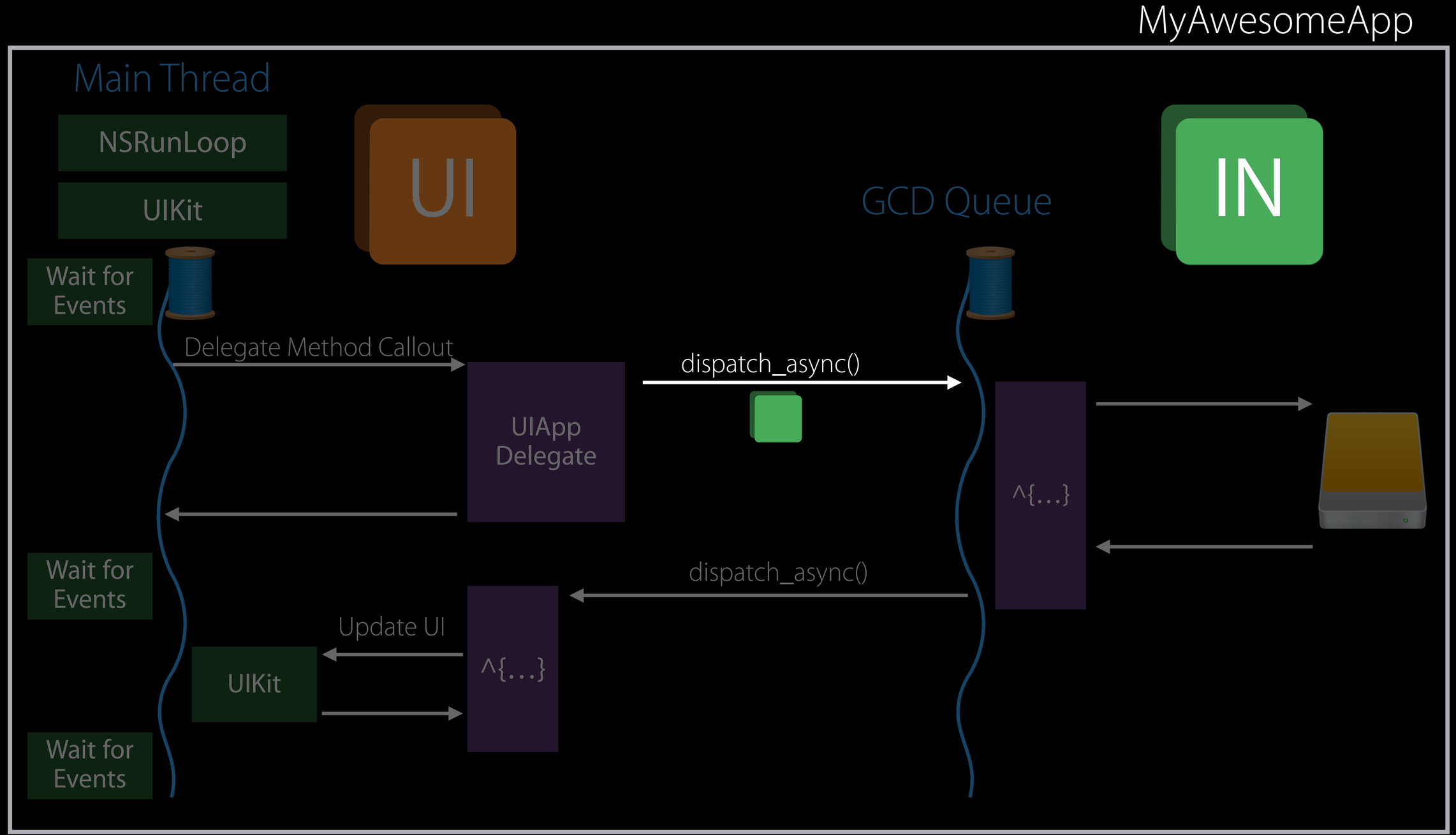
Asynchronous Work



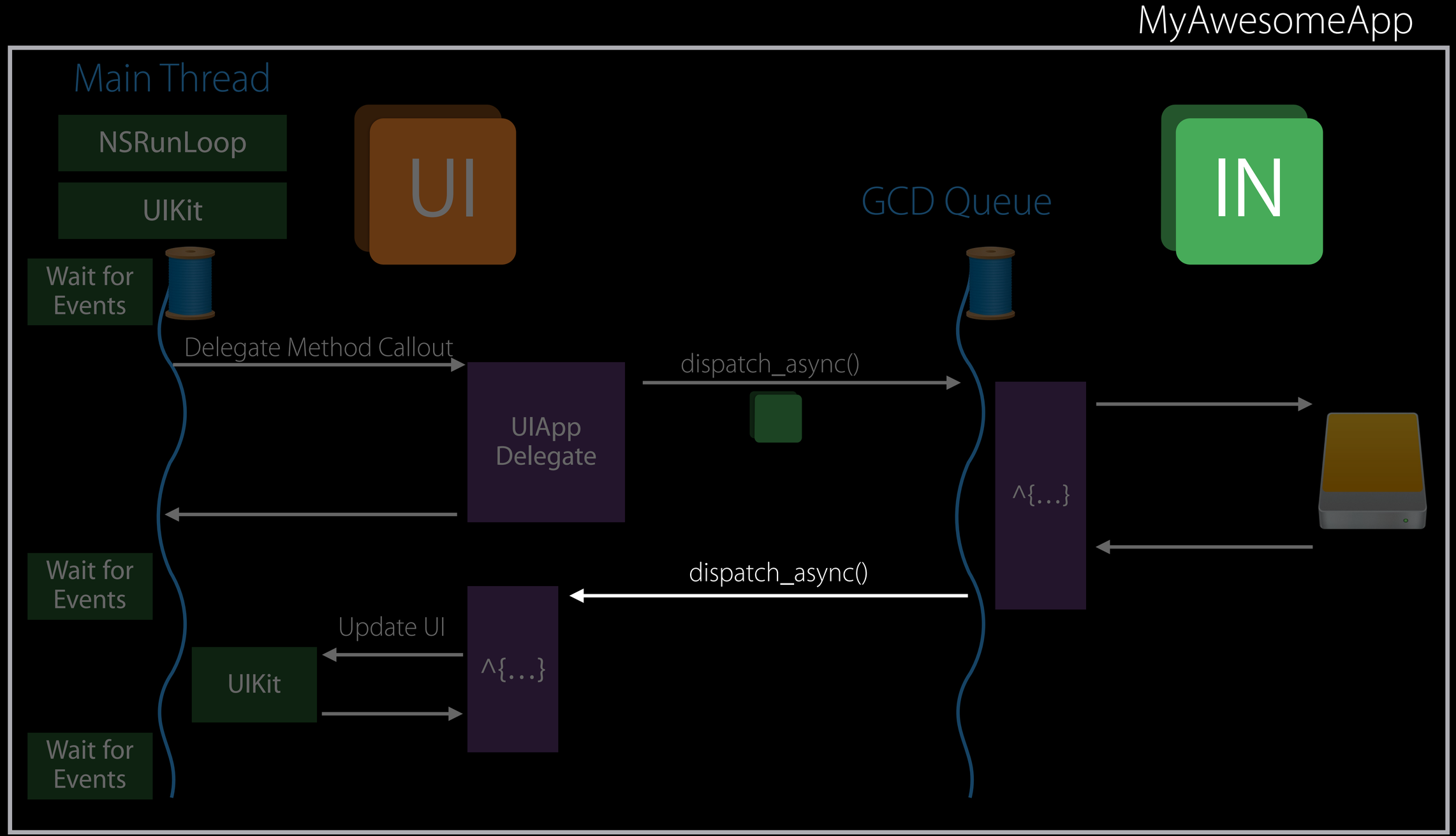
Asynchronous Work



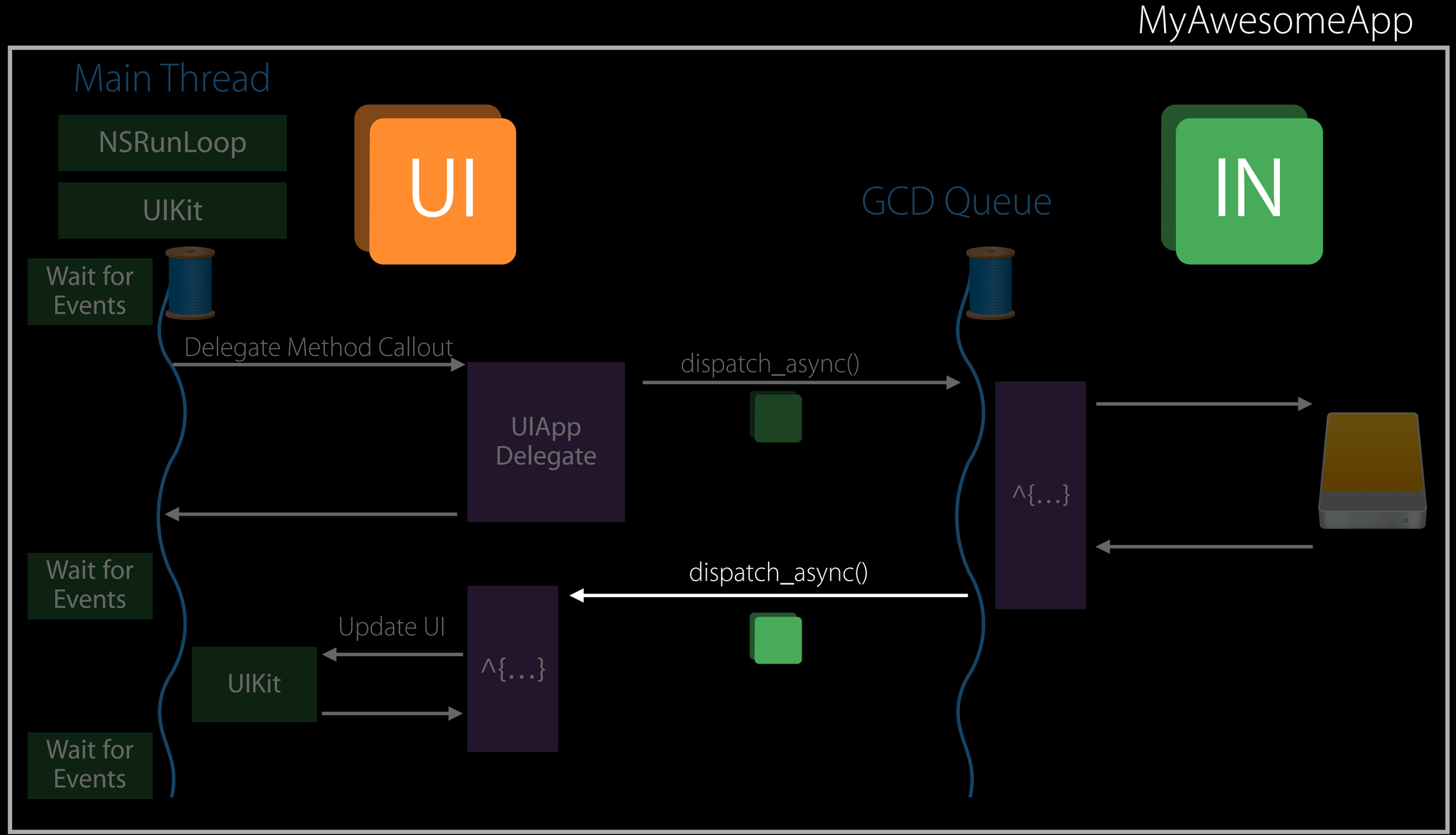
Asynchronous Work



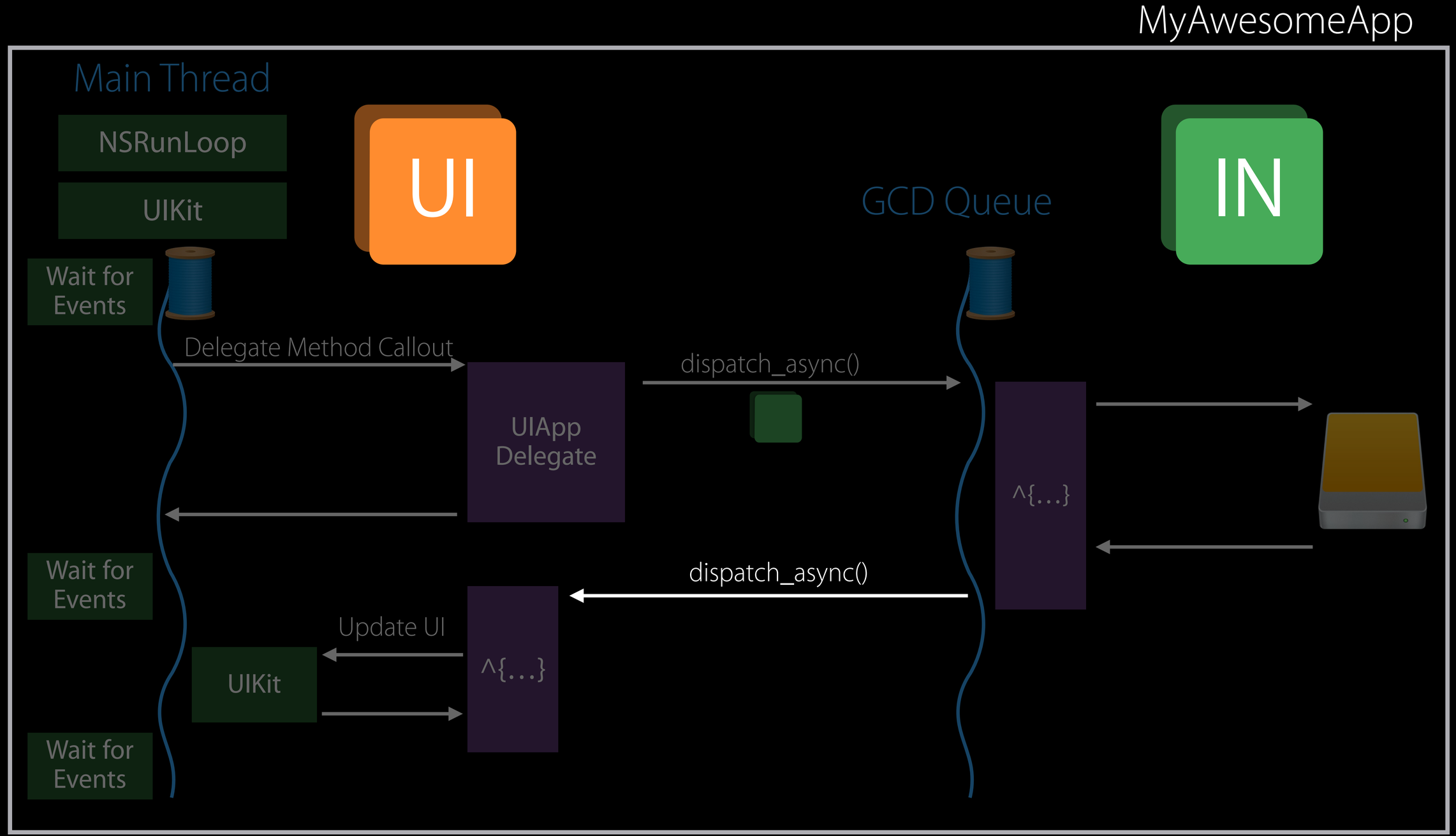
Asynchronous Work



Asynchronous Work



Asynchronous Work



QoS Propagation

Inferred QoS

QoS Propagation

Inferred QoS

QoS captured at the time of Block submission

- User Interactive translated to User Initiated

QoS Propagation

Inferred QoS

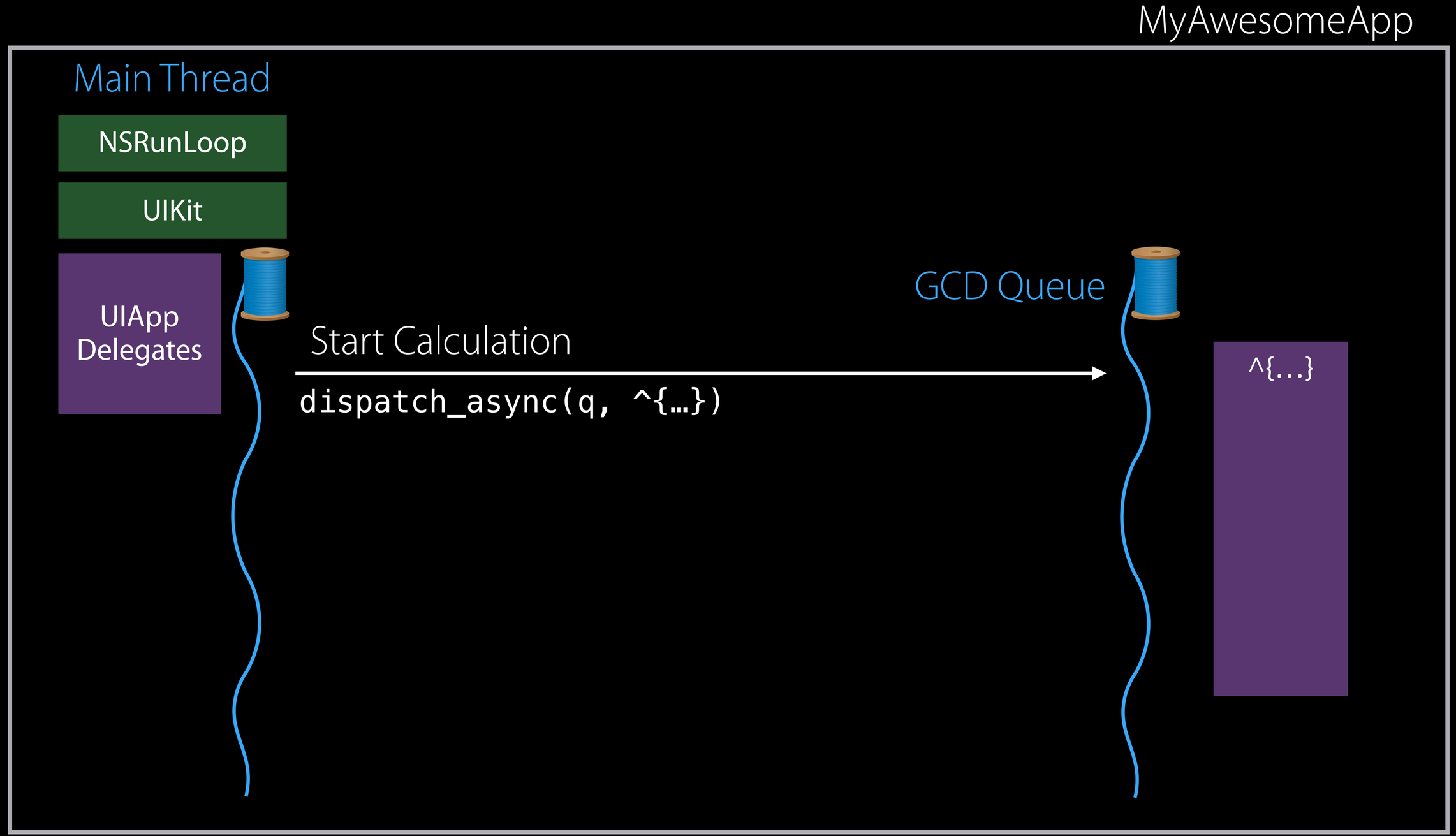
QoS captured at the time of Block submission

- User Interactive translated to User Initiated

Used if destination does not have QoS specified

- Does not lower QoS

Long-Running Job



Long-Running Job

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UIApp Delegates

GCD Queue

^{...}

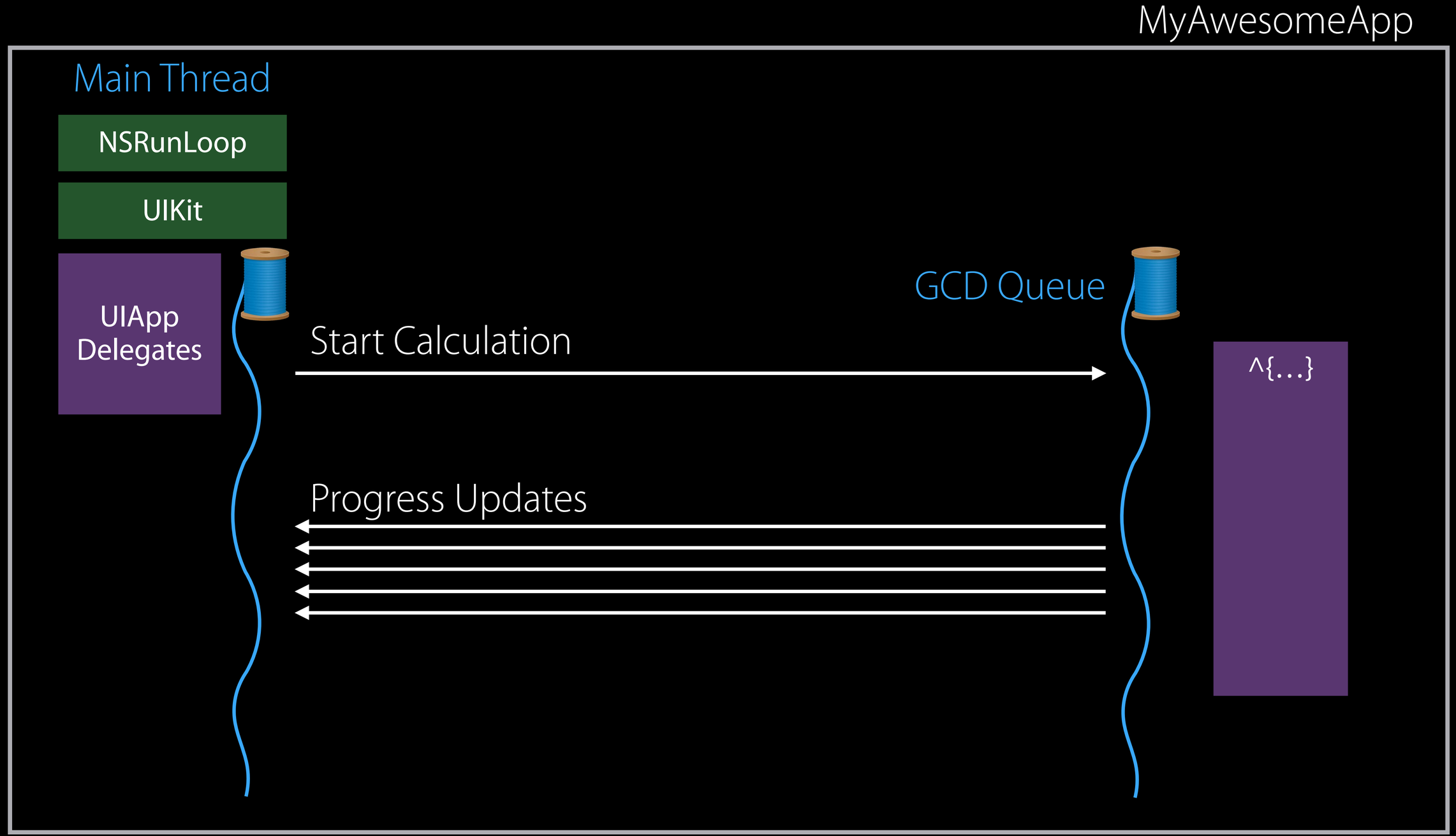
Start Calculation

```
dispatch_async(q, ^{…})
```

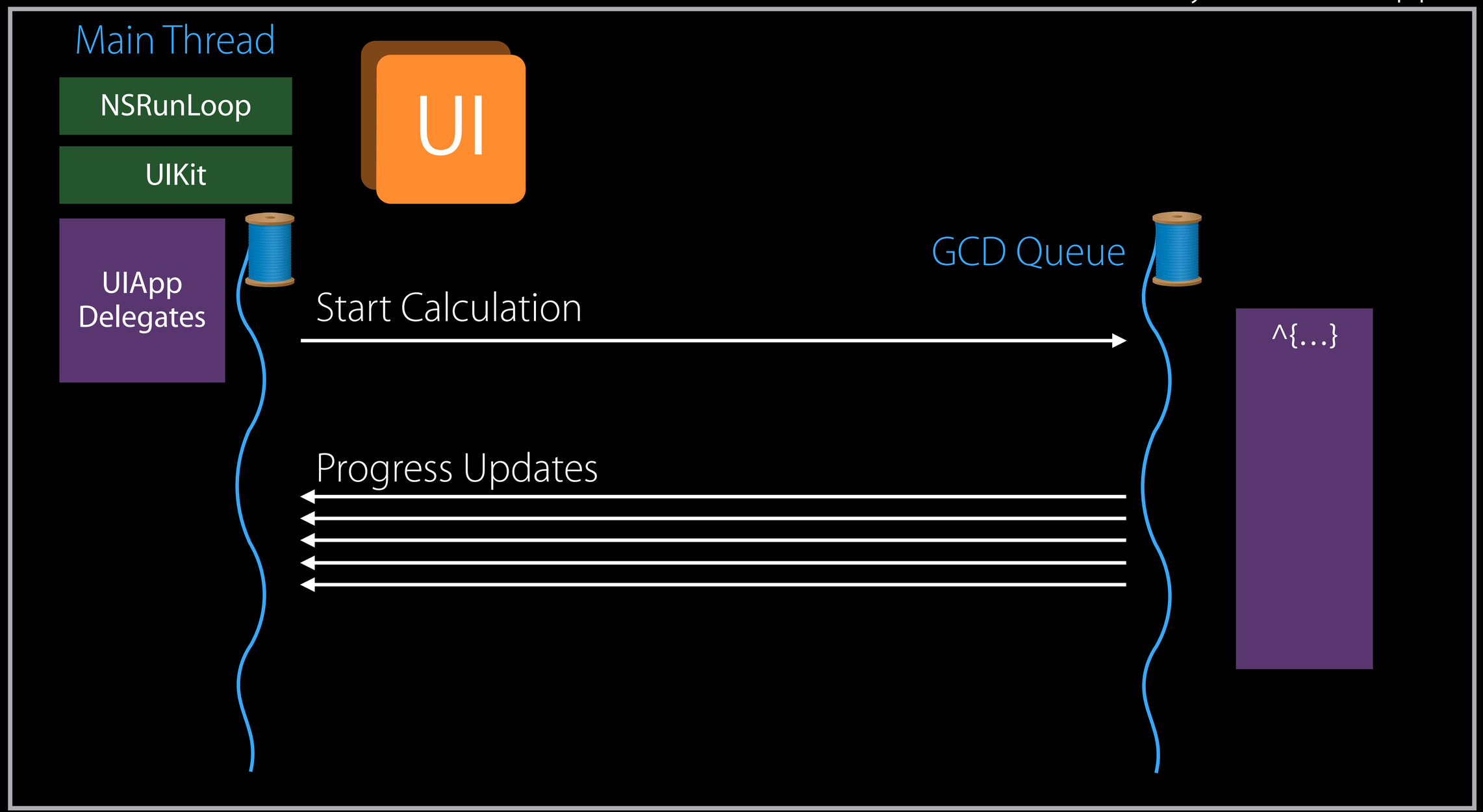
Progress Updates

```
dispatch_async(dispatch_get_main_queue(),^{
    // Update Progress UI
})
```

Long-Running Job



Long-Running Job



Long-Running Job

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UIApp Delegates

UI

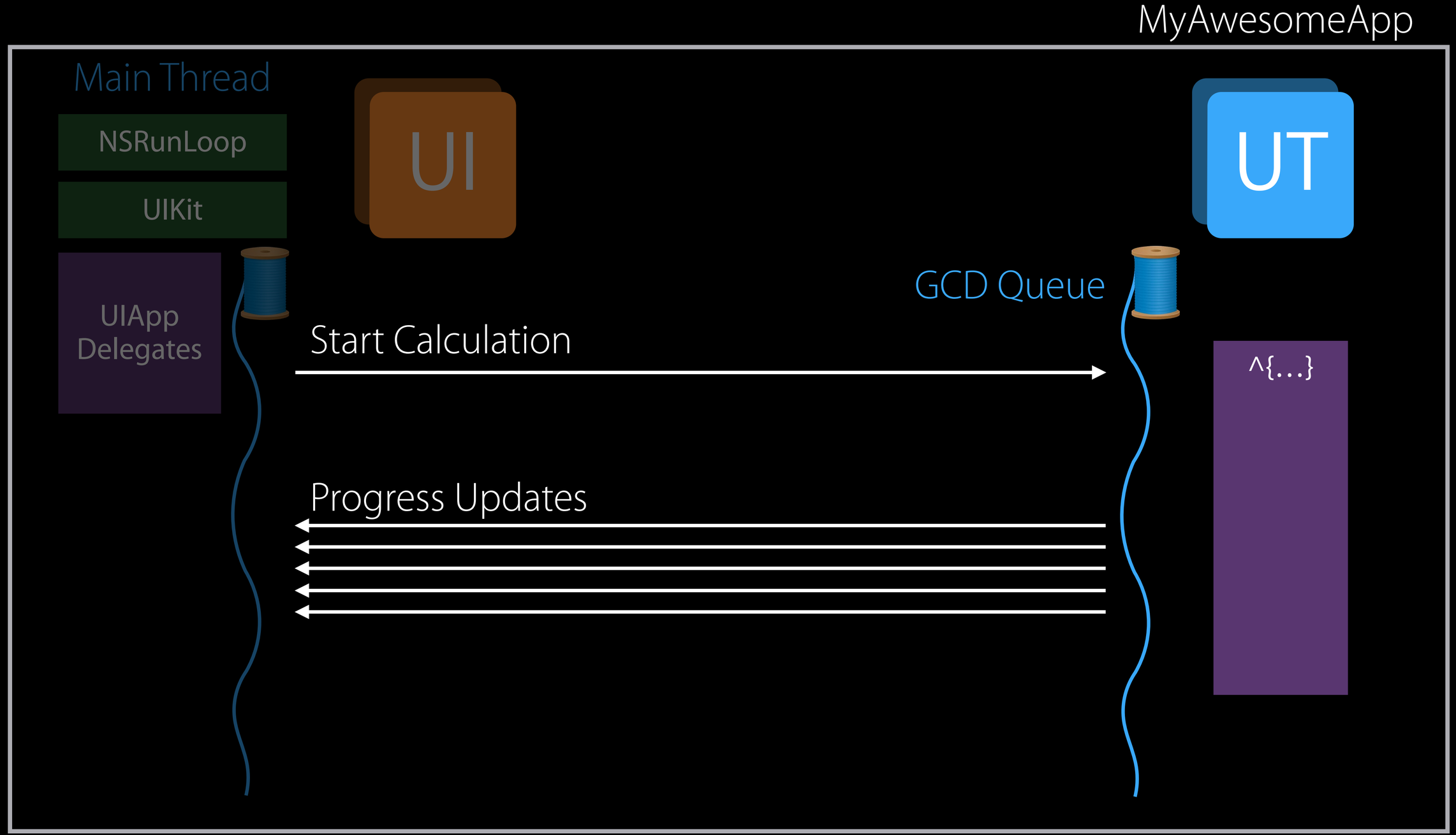
GCD Queue

^{...}

Start Calculation

Progress Updates

Long-Running Job



Long-Running Job

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UIApp Delegates

UI

Start Calculation

GCD Queue

UT

^{…}

Long-Running Job

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UIApp Delegates

UI

UT

GCD Queue

Start Calculation

```
b = dispatch_block_create_with_qos_class(
        0, QOS_CLASS_UTILITY, 0, ^{…})
dispatch_async(q, b)
```

^{...}

Long-Running Job

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UIApp Delegates

UI

UT

GCD Queue

Start Calculation

```
b = dispatch_block_create_with_qos_class(
        0, QOS_CLASS_UTILITY, 0, ^{…})
dispatch_async(q, b)
```

^{...}

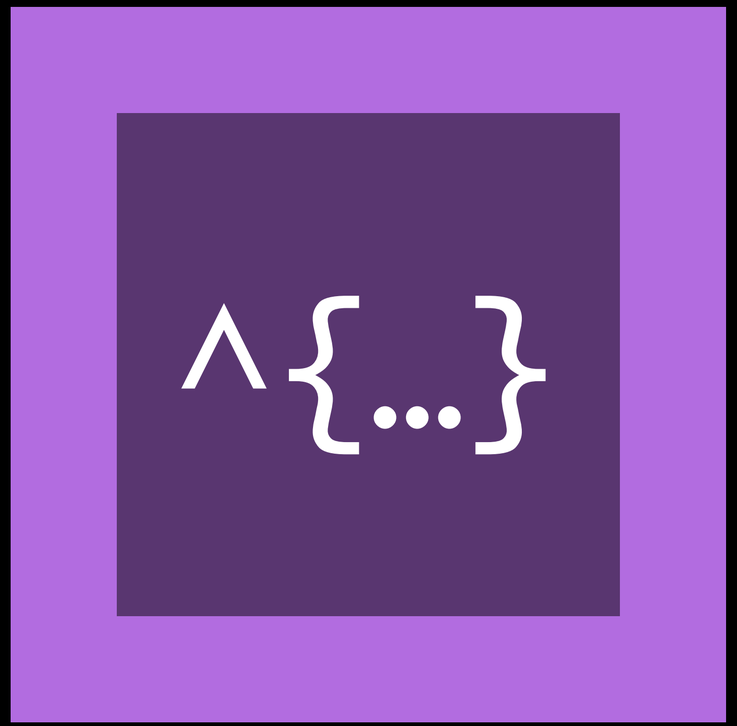
dispatch_async()

^{...}

Block QoS

Block created with explicit QoS attribute

- When work of another class is generated



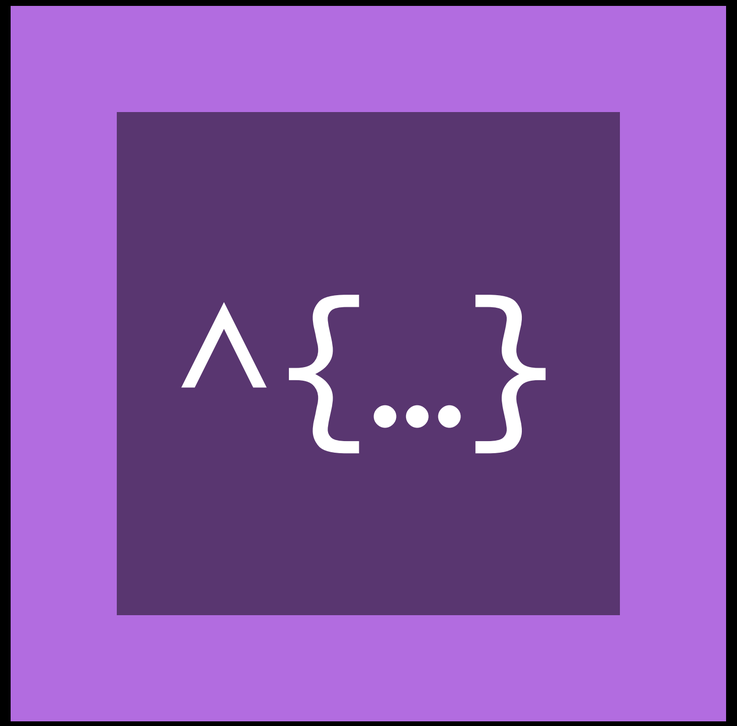
Block QoS

Block created with explicit QoS attribute

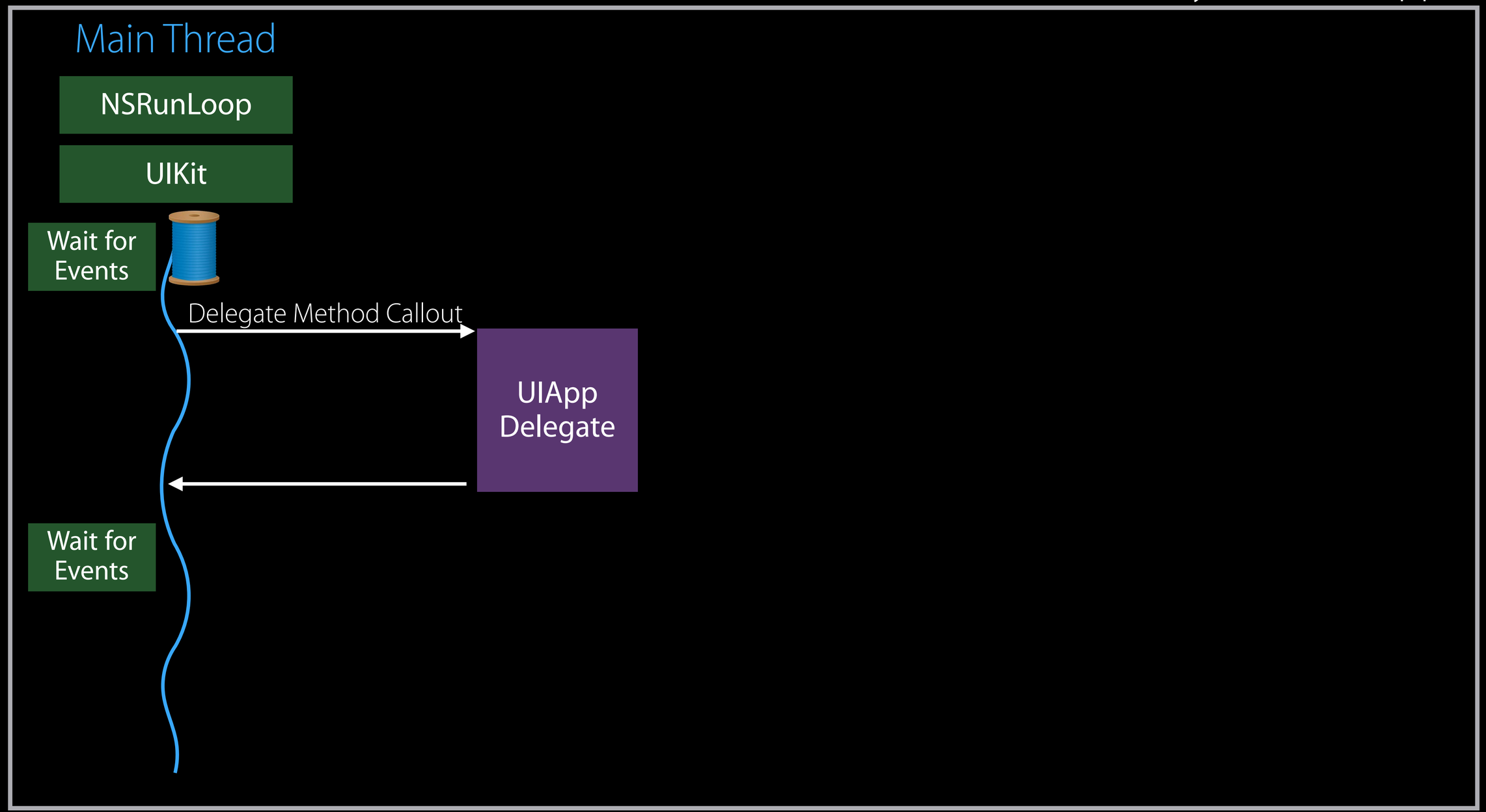
- When work of another class is generated

Captured at Block object creation

- DISPATCH_BLOCK_ASSIGN_CURRENT
- Store a callback Block for later submission



Maintenance Task



Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UI

Wait for Events

Delegate Method Callout

UIApp Delegate

Wait for Events

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

Delegate Method Callout

UIApp Delegate

Wait for Events

GCD Queue

dispatch_async()

^{...}

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

Delegate Method Callout

UIApp Delegate

Wait for Events

GCD Queue

dispatch_async()

^{…}

BG

Maintenance Task

MyAwesomeApp

Main Thread

```
qos_attr =
    dispatch_queue_attr_make_with_qos_class(
    attr, QOS_CLASS_BACKGROUND, 0)
q = dispatch_queue_create(“cleanup”, qos_attr)
```

Q

^{...}

BG

Wa
Events

Delegate Method Callout

dispatch_async()

UIApp
Delegate

Wait for
Events

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

Delegate Method Callout

UIApp Delegate

Wait for Events

GCD Queue

Q

^{…}

dispatch_async()

BG

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

Delegate Method Callout

UIApp Delegate

Wait for Events

GCD Queue

Q

^{…}

dispatch_async()

BG

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

Delegate Method Callout

UIApp Delegate

Wait for Events

GCD Queue

Q

^{…}

dispatch_async()

BG

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

GCD Queue

Q

^{…}

BG

Delegate Method Callout

dispatch_async()

```
dispatch_block_create(
    DISPATCH_BLOCK_DETACHED, ^{…})
```

Wait for Events

Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

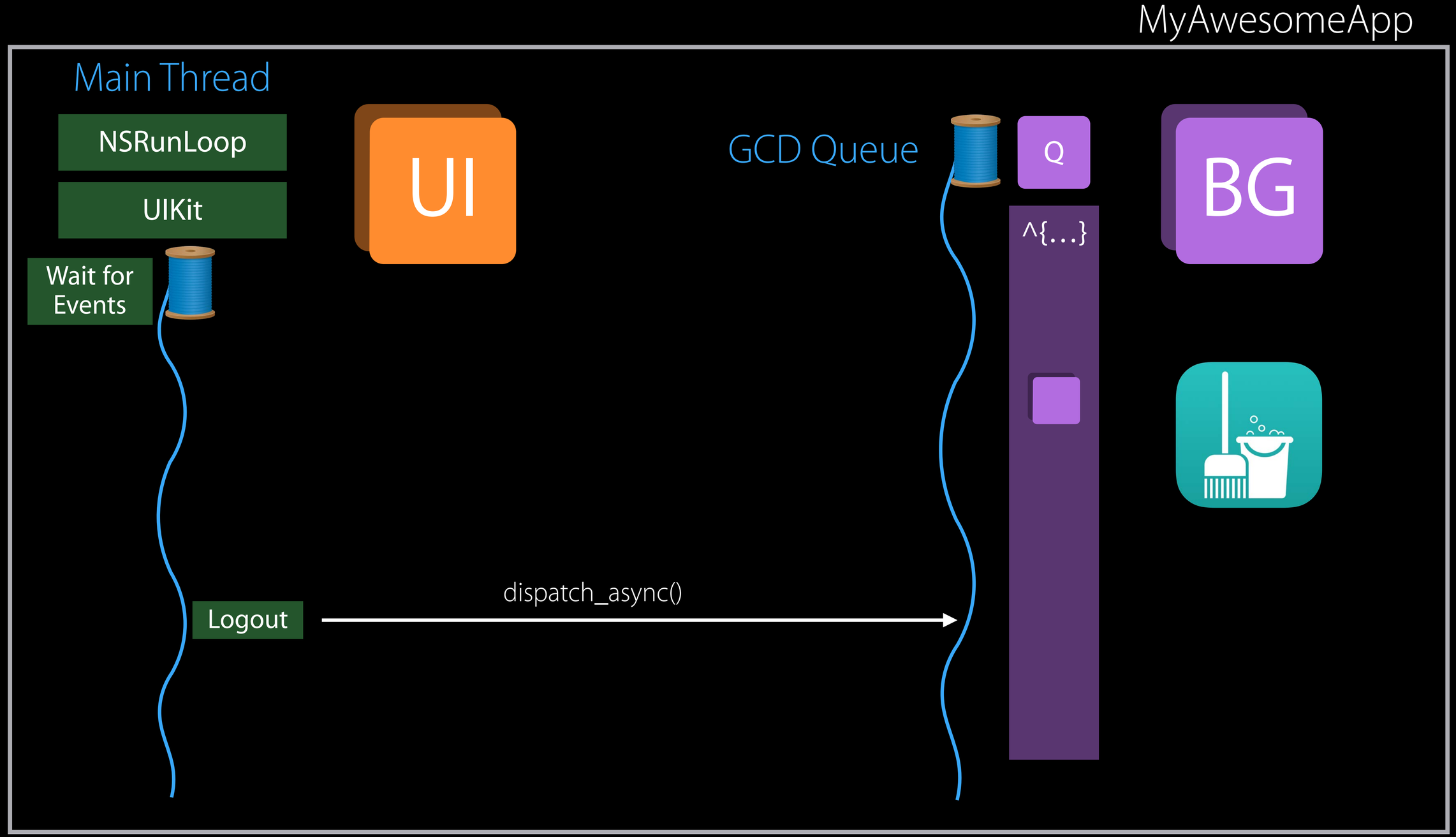
GCD Queue

Q

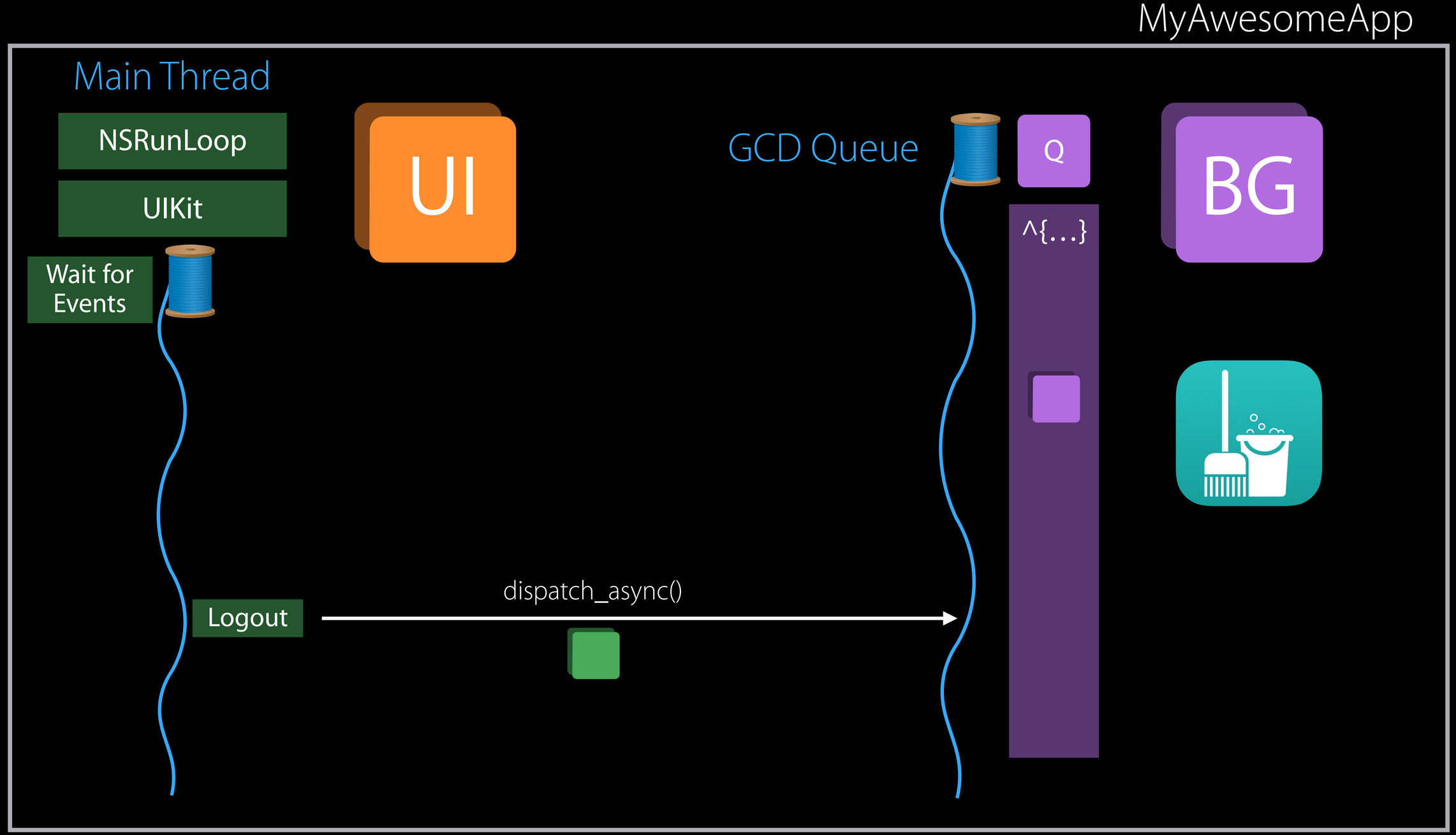
^{…}

BG

Maintenance Task



Maintenance Task



Maintenance Task

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

Wait for Events

UI

GCD Queue

Q

^{…}

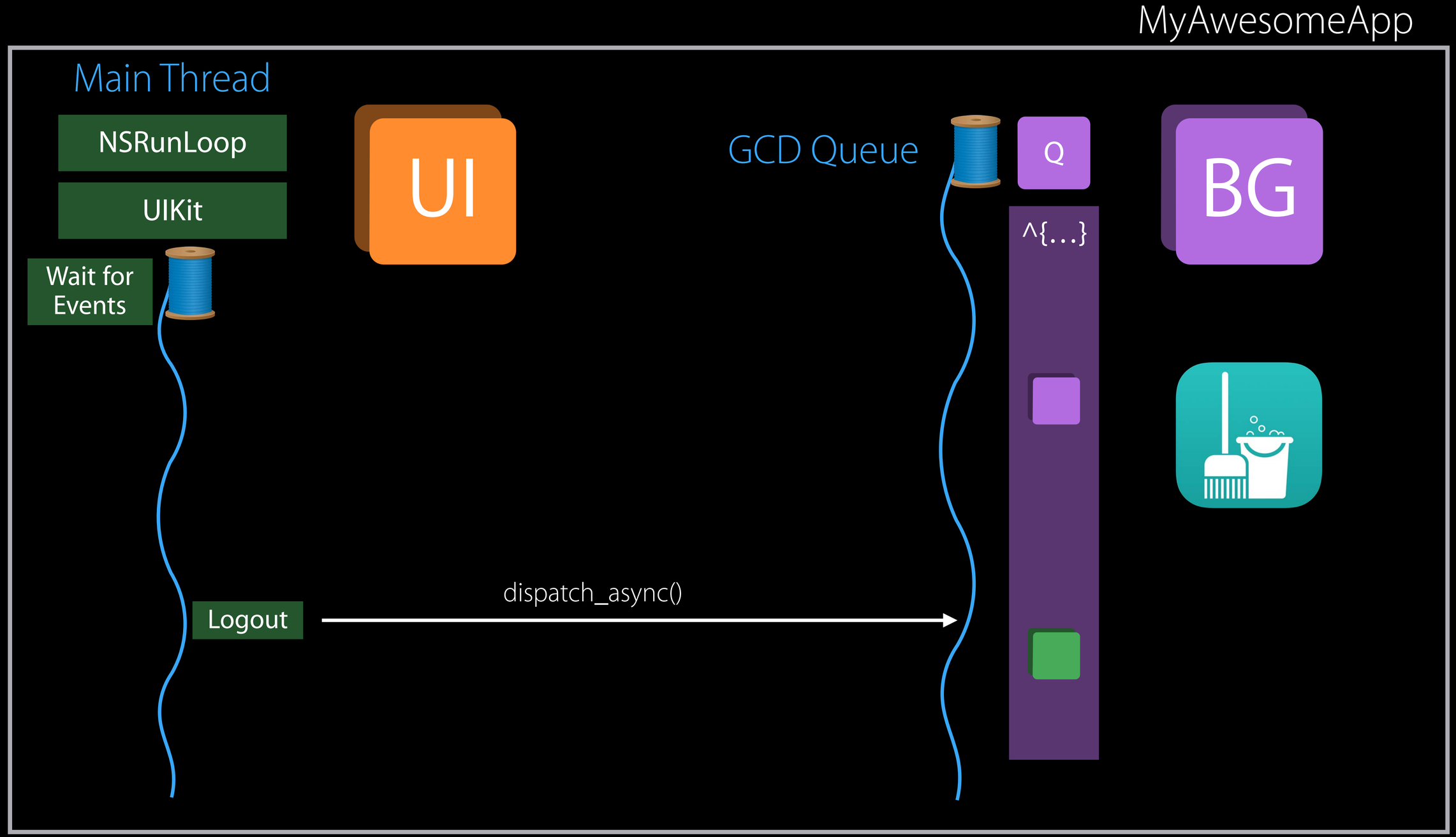
BG

```
dispatch_block_create(
    DISPATCH_BLOCK_ENFORCE_QOS_CLASS, ^{…})
```

dispatch_async()

Logout

Maintenance Task



Asynchronous Priority Inversion

High QoS Block submitted to serial queue

- Queue already contains Blocks with lower QoS

Asynchronous Priority Inversion

High QoS Block submitted to serial queue

- Queue already contains Blocks with lower QoS

QoS is raised until high QoS Block is reached

- Invisible to Blocks themselves

Queue QoS

Recap

Queues that are single purpose

- Detached Blocks may also be appropriate

Queue QoS

Recap

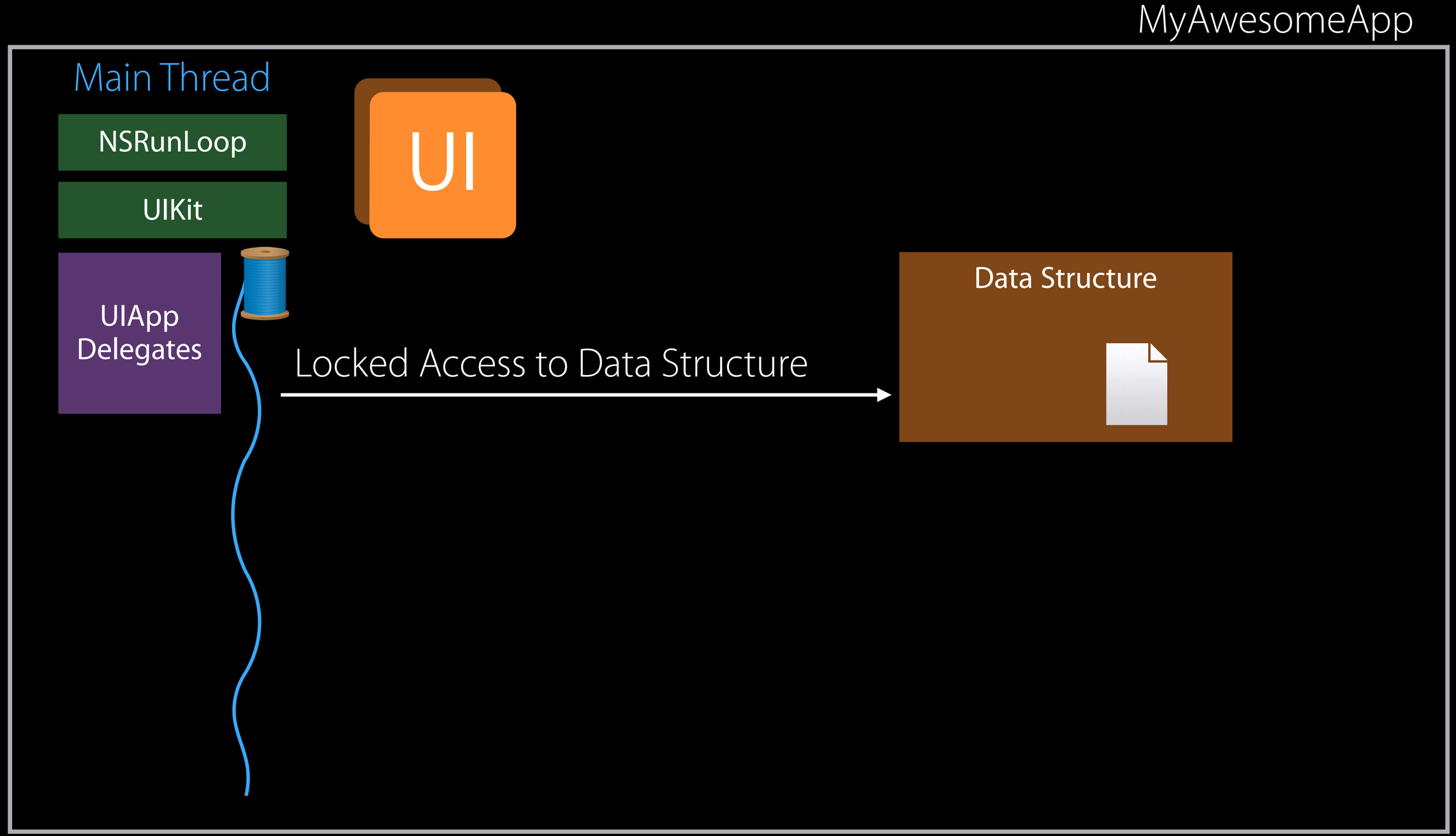
Queues that are single purpose

- Detached Blocks may also be appropriate

Ignore QoS in asynchronous Blocks

- Exceptional cases can enforce Block QoS

Queues as Locks



Queues as Locks

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UI

UIApp Delegates

Locked Access to Data Structure

Data Structure

Q

```
dispatch_queue_create(“com.example.data”,
                      DISPATCH_QUEUE_SERIAL)
```

Queues as Locks

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UI

UIApp Delegates

Locked Access to Data Structure

```
dispatch_sync(data.q, ^{…})
```

Data Structure

Q

Queues as Locks

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UI

UIApp Delegates

Locked Access to Data Structure

```
dispatch_sync(data.q, ^{…})
```

^{...}

Data Structure

Q

Queues as Locks

MyAwesomeApp

Main Thread

NSRunLoop

UIKit

UI

UIApp Delegates

Locked Access to Data Structure

```
dispatch_sync(data.q, ^{…})
```

^{...}

Data Structure

Q

dispatch_sync()

^{...}

UT

Synchronous Priority Inversion

High QoS thread waiting on lower QoS work

QoS of waited on work is raised for

- `dispatch_sync()` and `dispatch_block_wait()` of Blocks on serial queues
- pthread_mutex_lock()

Queues, Threads, and Run Loops

Anthony J. Chivetta Darwin Runtime Engineer

Your App

MyAwesomeApp

Main Thread

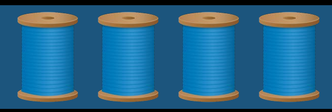
NSRunLoop

UIKit

UIApp Delegate

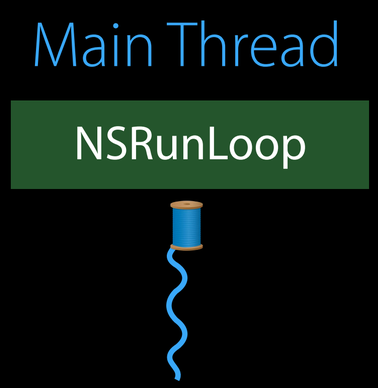
Q Q Q

GCD Thread Pool



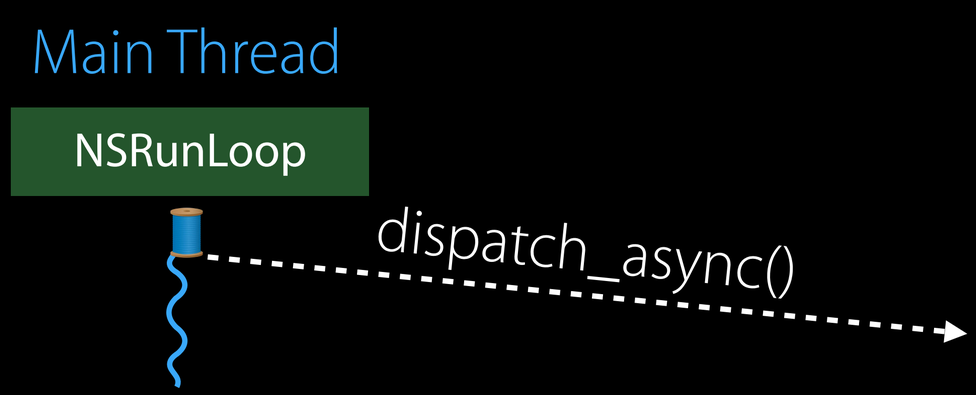
Run Loop Versus Queue

```
dispatch_async(q, ^{
 [self performSelector:@selector(thing) withObject:nil afterDelay:1];
});
```



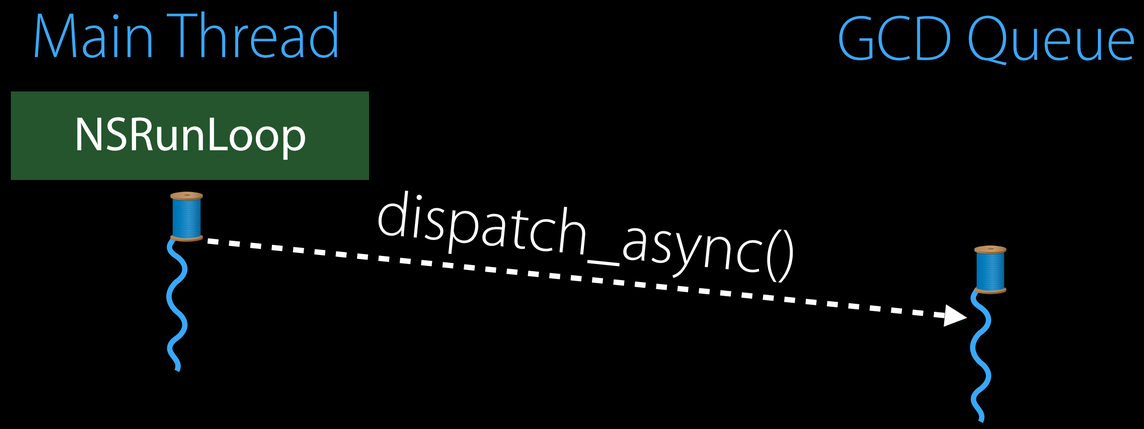
Run Loop Versus Queue

```
dispatch_async(q, ^{
 [self performSelector:@selector(thing) withObject:nil afterDelay:1];
});
```



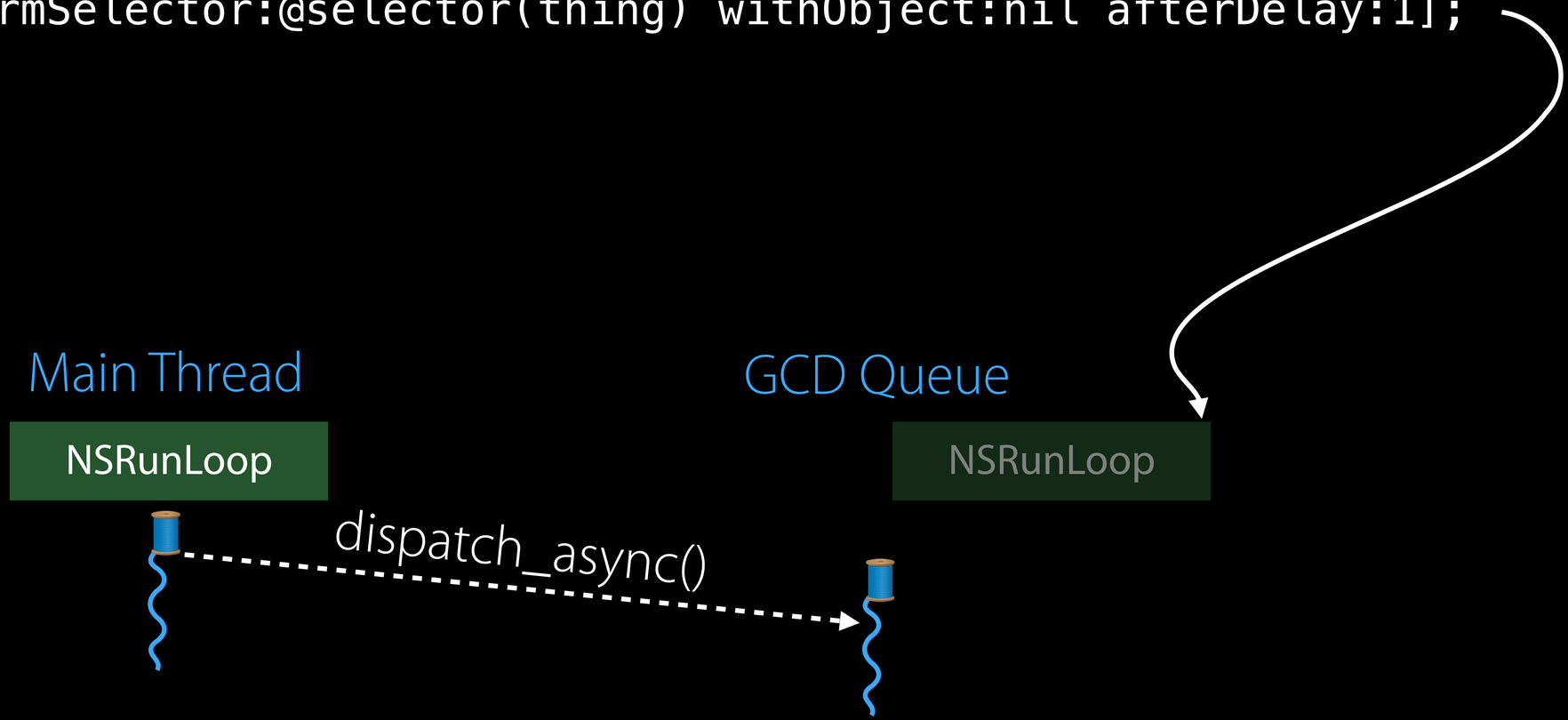
Run Loop Versus Queue

```
dispatch_async(q, ^{
 [self performSelector:@selector(thing) withObject:nil afterDelay:1];
});
```



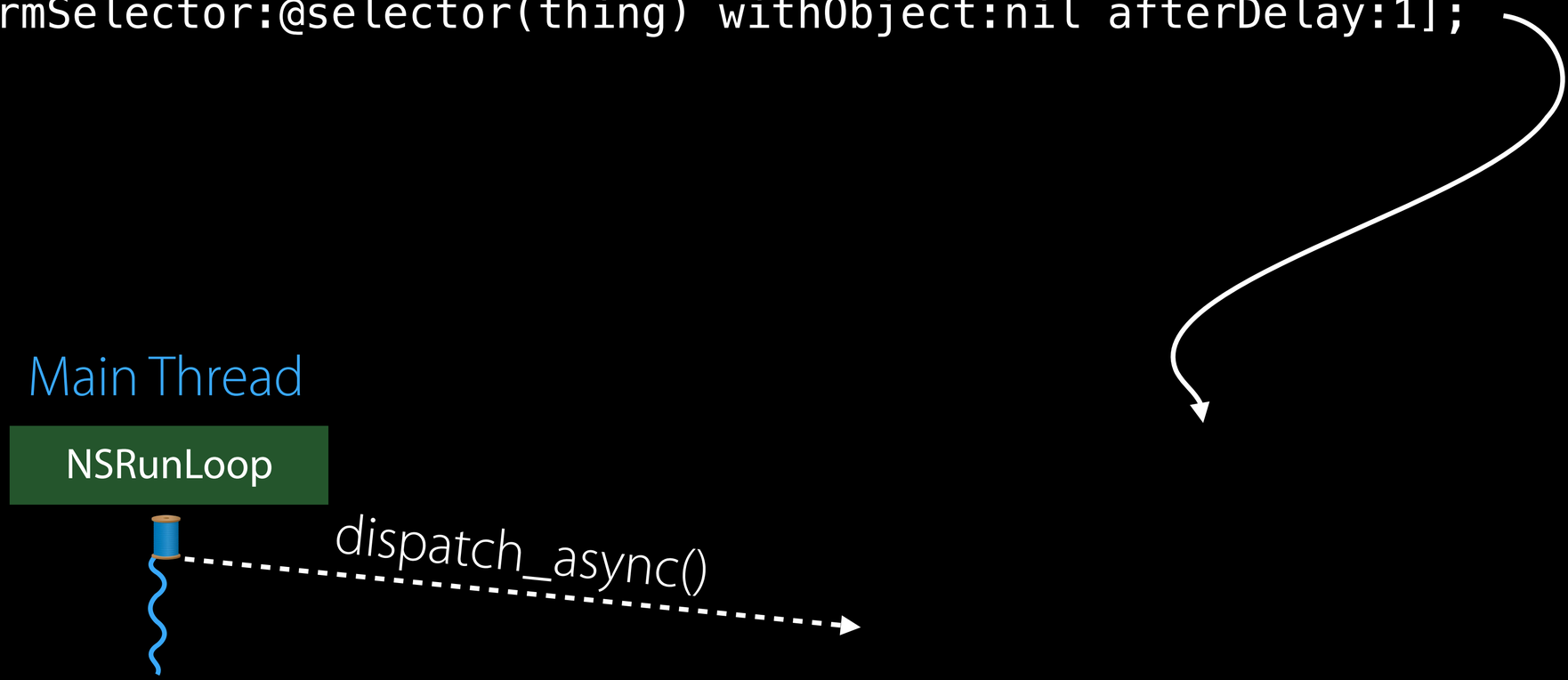
Run Loop Versus Queue

```
dispatch_async(q, ^{
 [self performSelector:@selector(thing) withObject:nil afterDelay:1];
});
```



Run Loop Versus Queue

```
dispatch_async(q, ^{
 [self performSelector:@selector(thing) withObject:nil afterDelay:1];
});
```



Run Loop Versus Queue

Properties

Run Loop

- Bound to a thread
- Gets delegate method callbacks
- Autorelease pool pops after each iteration
- Can be used reentrantly

Serial Queue

- Uses ephemeral threads
- Block callbacks
- Autorelease pool pops when thread idle
- Will deadlock if used reentrantly

The Main Thread's Run Loop is also exposed as the Main Queue

RunLoop Versus Queue

Timer APIs

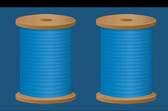
RunLoop

```
-[NSObject performSelector:withObject:afterDelay:]
+[NSTimer scheduledTimerWithTimeInterval:]
```

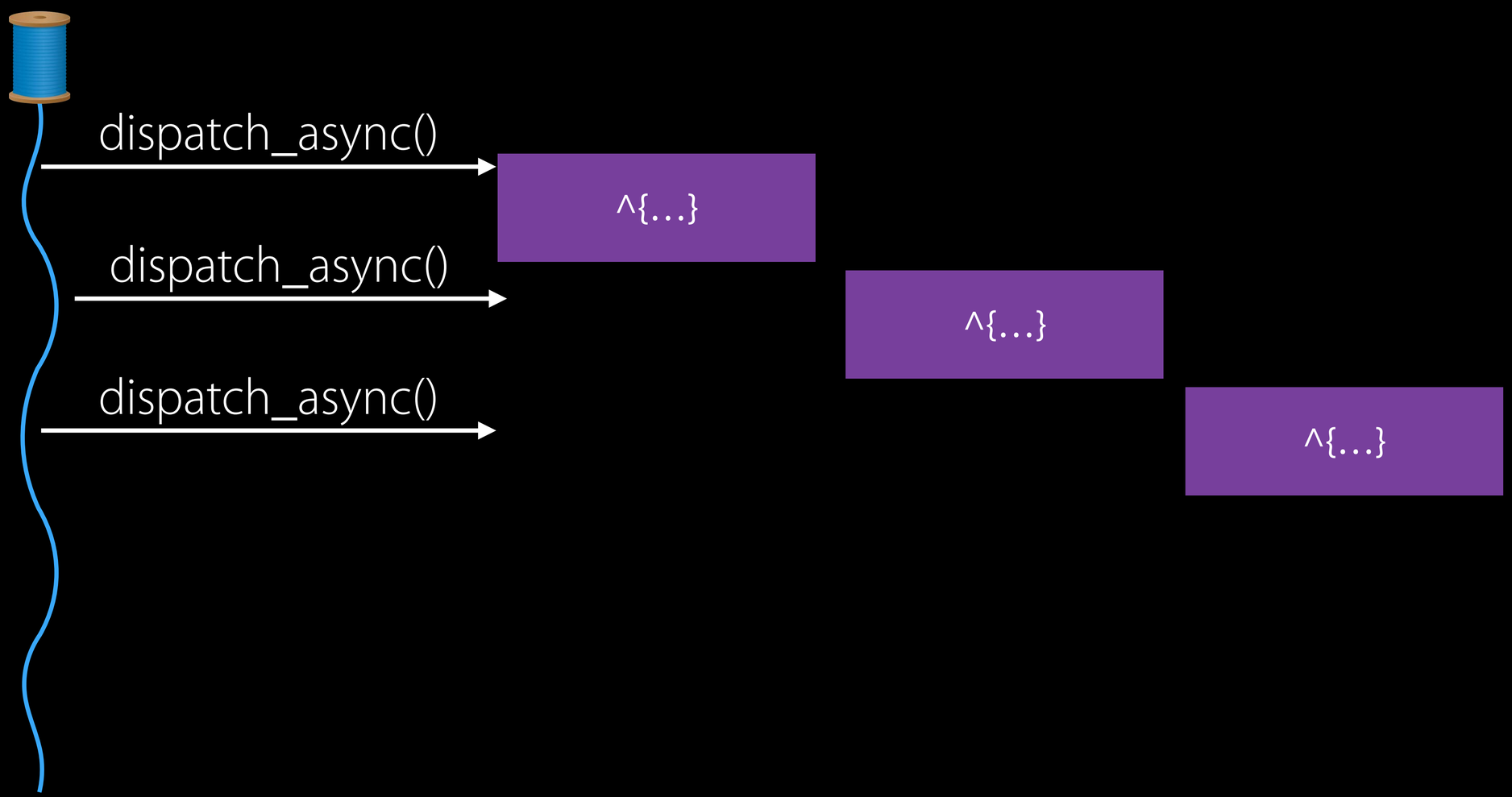
Queue

```
dispatch_after()
dispatch_source_set_timer()
```

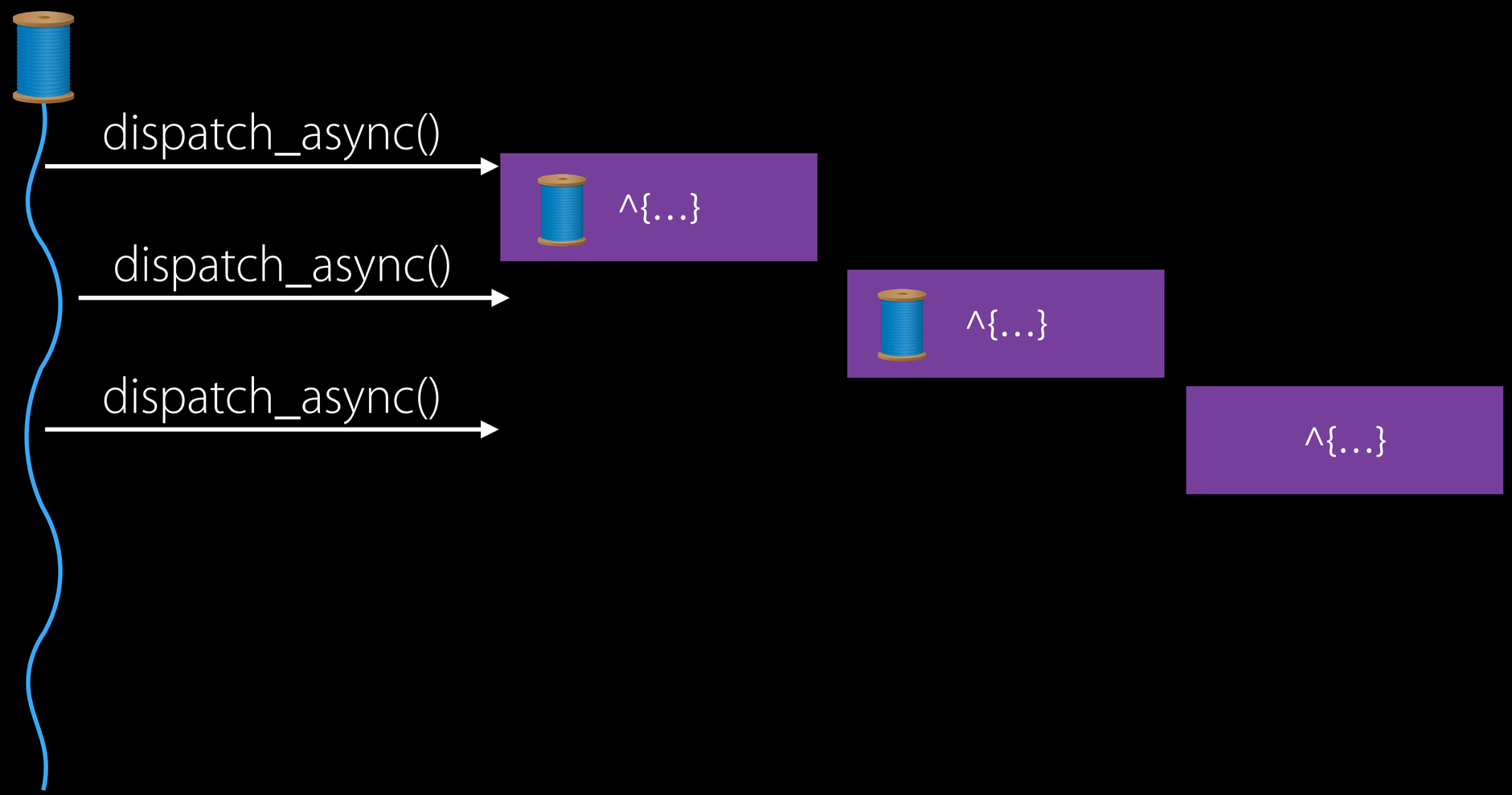
Thread Creation and Pooling

GCD Thread Pool 

Thread Creation and Pooling



Thread Creation and Pooling



Thread Creation and Pooling

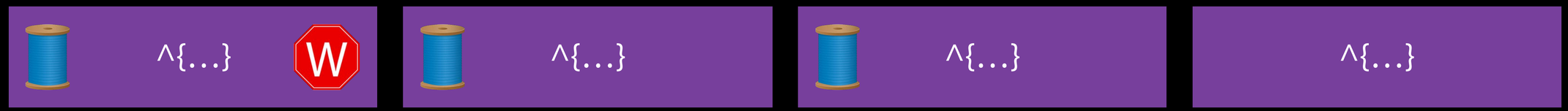
Waiting

A thread waits (blocks) when it needs to wait for a resource such as I/O or locks

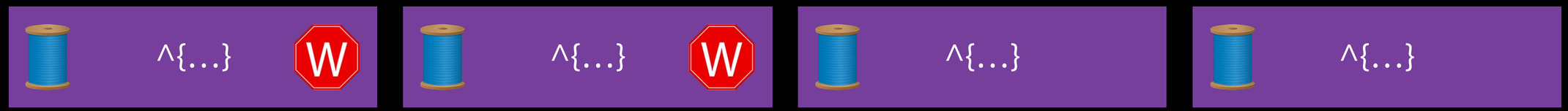
When a thread waits, GCD may spin up a new thread to ensure one thread per core

Thread Creation and Waiting

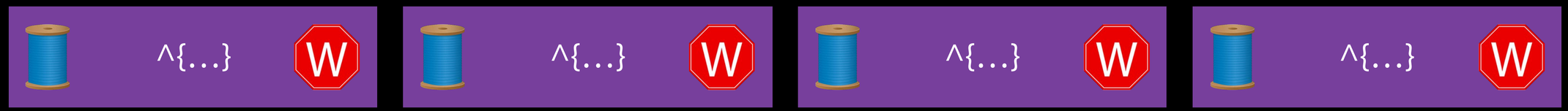
Thread Creation and Waiting



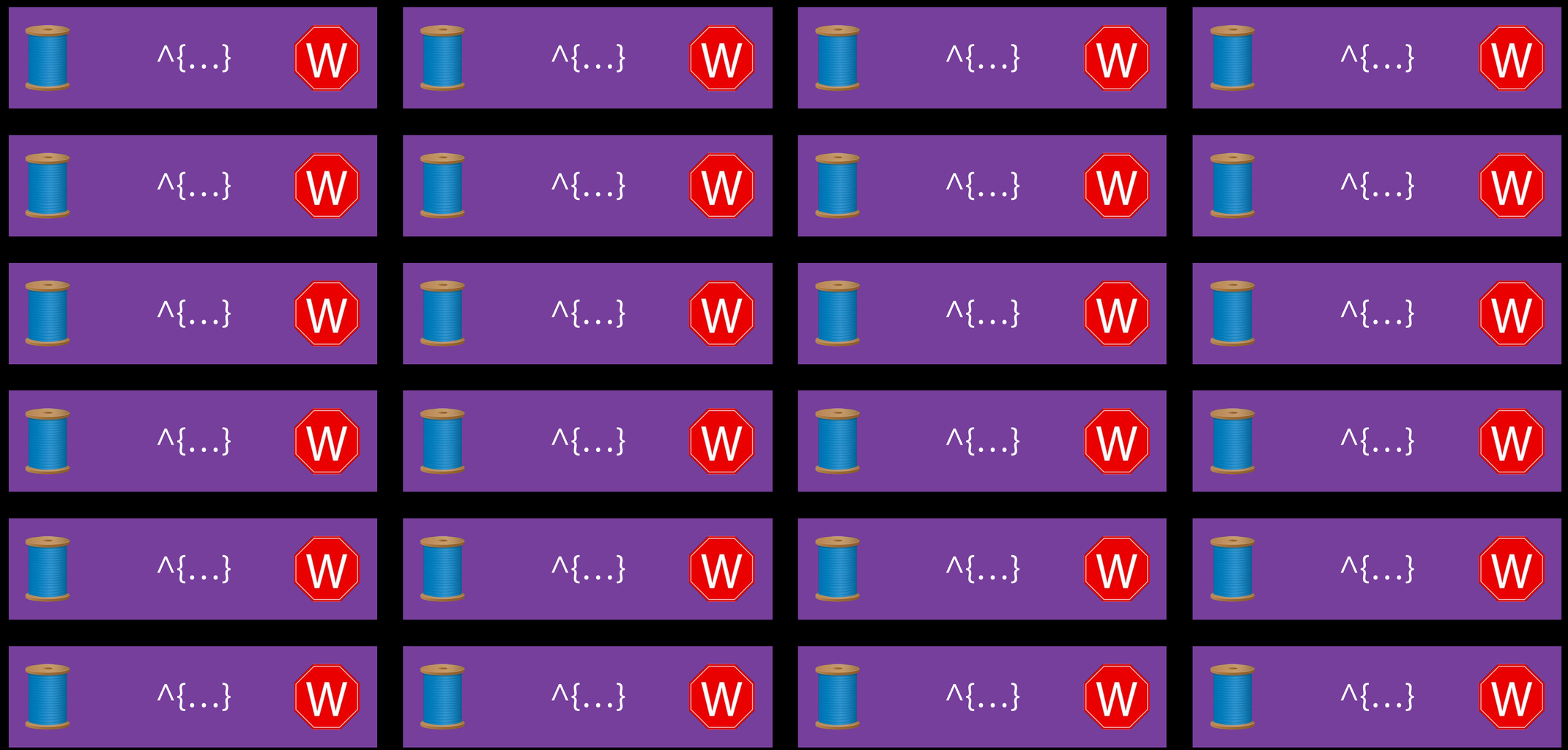
Thread Creation and Waiting



Thread Explosion



Thread Explosion



Thread Explosion Causing Deadlock

Main Thread

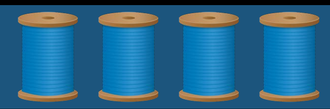
NSRunLoop

UIKit

Concurrent Queue

Serial Queue

GCD Thread Pool

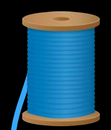


Thread Explosion Causing Deadlock

Main Thread

NSRunLoop

UIKit

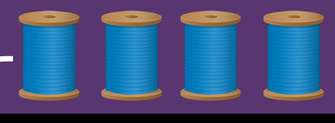


```
for (int i = 0; i < 999; i++) dispatch_async(q,^{…})
```

```
dispatch_sync(dispatch_get_main_queue(),^{…})
```

Concurrent Queue

LIMIT HIT



Serial Queue

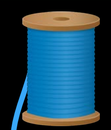
GCD Thread Pool

Thread Explosion Causing Deadlock

Main Thread

NSRunLoop

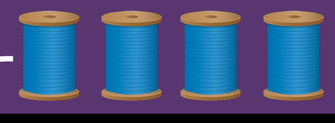
UIKit



```
for (int i = 0; i < 999; i++) dispatch_async(q,^{…})
```

Concurrent Queue

LIMIT HIT



```
dispatch_sync(dispatch_get_main_queue(),^{…})
```

```
dispatch_async(q,^{…})
```

Serial Queue

async Block

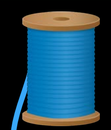
GCD Thread Pool

Thread Explosion Causing Deadlock

Main Thread

NSRunLoop

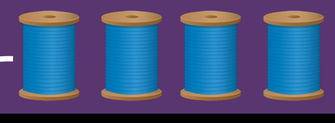
UIKit



```
for (int i = 0; i < 999; i++) dispatch_async(q,^{…})
```

Concurrent Queue

LIMIT HIT



```
dispatch_sync(dispatch_get_main_queue(),^{…})
```

```
dispatch_async(q,^{…})
```

```
dispatch_sync(q,^{…})
```

Serial Queue

sync Block

async Block

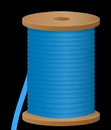
GCD Thread Pool

Thread Explosion Causing Deadlock

Main Thread

NSRunLoop

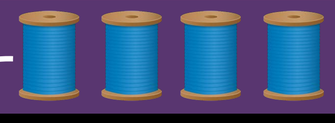
UIKit



```
for (int i = 0; i < 999; i++) dispatch_async(q,^{…})
```

Concurrent Queue

LIMIT HIT



```
dispatch_sync(dispatch_get_main_queue(),^{…})
```

```
dispatch_async(q,^{…})
```

```
dispatch_sync(q,^{…})
```

Serial Queue

sync Block

No Threads!

GCD Thread Pool

Avoiding Thread Explosion

Always good advice: use asynchronous APIs, especially for I/O

Use serial queues

Use NSOperationQueues with concurrency limits

```
NSOperationQueue.maxConcurrentOperationCount
```

Don’t generate unlimited work…

Avoiding Thread Explosion

Mixing sync and async

```
// fast, just a lock
dispatch_sync(q, ^{…});

// fast, just an atomic enqueue
dispatch_async(q, ^{…});
// slow, has to wait far a thread to complete above Block
dispatch_sync(q, ^{…});
```

Be super careful about mixing these from the main thread!

Avoiding Thread Explosion

dispatch_apply

```
// DANGEROUS – may cause thread explosion and deadlocks
for (int i = 0; i < 999; i++){
    dispatch_async(q, ^{…});
}
dispatch_barrier_sync(q, ^{});
```

```
// GOOD – GCD will manage parallelism
dispatch_apply(999, q, ^(size_t i){…});
```

Avoiding Thread Explosion

dispatch_semaphore

```
#define CONCURRENT_TASKS 4
sema = dispatch_semaphore_create(CONCURRENT_TASKS);
for (int i = 0; i < 999; i++){
    dispatch_async(q, ^{
        // do work
        dispatch_semaphore_signal(sema);
    });
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
}
```

GCD and Crash Reports

```
Process:             Game [568]
Path:                /Applications/2048 Game.app/Contents/MacOS/Game
Identifier:          com.examples
Version:             1.5 (1.5)
App Item ID:         871033113
App External ID:     655842710
Code Type:           X86-64 (Native)
Parent Process:      ??? [1]
Responsible:         2048 Game [568]
User ID:             501

Date/Time:           2015-06-10 15:17:18.745 -0700
OS Version:          Mac OS X 10.11 (15A199)
Report Version:      11
Anonymous UUID:      B5CC6B16-514B-067B-E1C4-5FBFA9BEFEF8

Sleep/Wake UUID:     B9324D7F-B36A-4495-A750-F2D5E0476B85

Time Awake Since Boot: 11000 seconds

Crashed Thread:      0  Dispatch queue: com.apple.main-thread

Exception Type:      EXC_CRASH (SIGABRT)
Exception Codes:     0x0000000000000000, 0x0000000000000000
Exception Note:      EXC_CORPSE_NOTIFY

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_kernel.dylib              0x00007fff896779c6 mach_msg_trap + 10
1   libsystem_kernel.dylib              0x00007fff89676e07 mach_msg + 55
2   com.apple.CoreFoundation            0x00007fff8c862004 __CFRunLoopServiceMachPort + 212
3   com.apple.CoreFoundation            0x00007fff8c8614cc __CFRunLoopRun + 1356
4   com.apple.CoreFoundation            0x00007fff8c860d18 CFRunLoopRunSpecific + 296
5   com.apple.HIToolbox                 0x00007fff9081e52d RunCurrentEventLoopInMode + 235
6   com.apple.HIToolbox                 0x00007fff9081e2c3 ReceiveNextEventCommon + 432
7   com.apple.HIToolbox                 0x00007fff9081e103 _BlockUntilNextEventMatchingListInModeWithFilter + 71
8   com.apple.AppKit                    0x00007fff8e4aa70e _DPSNextEvent + 927
9   com.apple.AppKit                    0x00007fff8e87882d -[NSApplication _nextEventMatchingEventMask:untilDate:inMode:dequeue:] + 324
10  com.apple.AppKit                    0x00007fff8e4a069f -[NSApplication run] + 682
11  com.apple.AppKit                    0x00007fff8e422bb3 NSApplicationMain + 1176
12  libdyld.dylib                       0x00007fff8c56a5ad start + 1

Thread 1:: Dispatch queue: com.apple.libdispatch-manager
0   libsystem_kernel.dylib              0x00007fff8967e08a kevent_qos + 10
1   libdispatch.dylib                   0x00007fff8be05811 _dispatch_mgr_invoke + 251
2   libdispatch.dylib                   0x00007fff8be05465 _dispatch_mgr_thread + 52

Thread 2:
0   libsystem_kernel.dylib              0x00007fff8967d29a __semwait_signal + 10
1   libsystem_c.dylib                   0x00007fff982c2b05 nanosleep + 199
2   libc++.1.dylib                      0x00007fff89f10100 std::__1::this_thread::sleep_for(std::__1::chrono::duration<long long, std::__1::ratio<1l, 1000000000l> > const&) + 75
3   com.apple.JavaScriptCore            0x00007fff8dd1cde7 bmalloc::Heap::scavenge(std::__1::unique_lock<bmalloc::StaticMutex>&, std::__1::chrono::duration<long long, std::__1::ratio<1l, 1000l> >) + 375
4   com.apple.JavaScriptCore            0x00007fff8dd1c964 bmalloc::Heap::concurrentScavenge() + 68
5   com.apple.JavaScriptCore            0x00007fff8dd1ef0a bmalloc::AsyncTask<bmalloc::Heap, void (bmalloc::Heap::*)()>::entryPoint() + 90
6   com.apple.JavaScriptCore            0x00007fff8dd1eea9 bmalloc::AsyncTask<bmalloc::Heap, void (bmalloc::Heap::*)()>::pthreadEntryPoint(void*) + 9
7   libsystem_pthread.dylib             0x00007fff8fd31cc3 _pthread_body + 131
8   libsystem_pthread.dylib             0x00007fff8fd31c40 _pthread_start + 168
9   libsystem_pthread.dylib             0x00007fff8fd2eda5 thread_start + 13

Thread 3:: com.apple.NSURLConnectionLoader
0   libsystem_kernel.dylib              0x00007fff896779c6 mach_msg_trap + 10
1   libsystem_kernel.dylib              0x00007fff89676e07 mach_msg + 55
2   com.apple.CoreFoundation            0x00007fff8c862004 __CFRunLoopServiceMachPort + 212
3   com.apple.CoreFoundation            0x00007fff8c8614cc __CFRunLoopRun + 1356
4   com.apple.CoreFoundation            0x00007fff8c860d18 CFRunLoopRunSpecific + 296
5   com.apple.CFNetwork                 0x00007fff8a0fa49a +[NSURLConnection(Loader) _resourceLoadLoop:] + 412
6   com.apple.Foundation                0x00007fff96a213c6 __NSThread__start__ + 1331
7   libsystem_pthread.dylib             0x00007fff8fd31cc3 _pthread_body + 131
8   libsystem_pthread.dylib             0x00007fff8fd31c40 _pthread_start + 168
9   libsystem_pthread.dylib             0x00007fff8fd2eda5 thread_start + 13

Thread 4:: com.apple.NSEventThread
0   libsystem_kernel.dylib              0x00007fff896779c6 mach_msg_trap + 10
1   libsystem_kernel.dylib              0x00007fff89676e07 mach_msg + 55
2   com.apple.CoreFoundation            0x00007fff8c862004 __CFRunLoopServiceMachPort + 212
3   com.apple.CoreFoundation            0x00007fff8c8614cc __CFRunLoopRun + 1356
4   com.apple.CoreFoundation            0x00007fff8c860d18 CFRunLoopRunSpecific + 296
5   com.apple.AppKit                    0x00007fff8e56c735 _NSEventThread + 149
6   libsystem_pthread.dylib             0x00007fff8fd31cc3 _pthread_body + 131
7   libsystem_pthread.dylib             0x00007fff8fd31c40 _pthread_start + 168
8   libsystem_pthread.dylib             0x00007fff8fd2eda5 thread_start + 13

Thread 5:: com.apple.CFSocket.private
0   libsystem_kernel.dylib              0x00007fff8967d20a __select + 10
1   com.apple.CoreFoundation            0x00007fff8c8adb8a __CFSocketManager + 762
2   libsystem_pthread.dylib             0x00007fff8fd31cc3 _pthread_body + 131
3   libsystem_pthread.dylib             0x00007fff8fd31c40 _pthread_start + 168
4   libsystem_pthread.dylib             0x00007fff8fd2eda5 thread_start + 13
```

Reading the Tea Leaves

Manager thread

```
Thread 1:: Dispatch queue: com.apple.libdispatch-manager
0   libsystem_kernel.dylib      0x00007fff8967e08a kevent_qos + 10
1   libdispatch.dylib           0x00007fff8be05811 _dispatch_mgr_invoke + 251
2   libdispatch.dylib           0x00007fff8be05465 _dispatch_mgr_thread + 52
```

Reading the Tea Leaves

Idle GCD thread

```
Thread 6:
0   libsystem_kernel.dylib      0x00007fff8967d772 __workq_kernreturn + 10
1   libsystem_pthread.dylib     0x00007fff8fd317d9 _pthread_wqthread + 1283
2   libsystem_pthread.dylib     0x00007fff8fd2ed95 start_wqthread + 13
```

Reading the Tea Leaves

Active GCD thread

```
Thread 3 Crashed:: Dispatch queue: <queue name>
                                          <my code>
7   libdispatch.dylib         0x07fff8fcfd323 _dispatch_call_block_and_release
8   libdispatch.dylib         0x07fff8fcf8c13 _dispatch_client_callout + 8
9   libdispatch.dylib         0x07fff8fcfc365 _dispatch_queue_drain + 1100
10  libdispatch.dylib         0x07fff8fcfdecc _dispatch_queue_invoke + 202
11  libdispatch.dylib         0x07fff8fcfb6b7 _dispatch_root_queue_drain + 463
12  libdispatch.dylib         0x07fff8fd09fe4 _dispatch_worker_thread3 + 91
13  libsystem_pthread.dylib   0x07fff93c17637 _pthread_wqthread + 729
14  libsystem_pthread.dylib   0x07fff93c1540d start_wqthread + 13
```

Reading the Tea Leaves

Idle main thread

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_kernel.dylib     0x00007fff906614de mach_msg_trap + 10
1   libsystem_kernel.dylib     0x00007fff9066064f mach_msg + 55
2   com.apple.CoreFoundation   0x00007fff9a8c1eb4 __CFRunLoopServiceMachPort
3   com.apple.CoreFoundation   0x00007fff9a8c137b __CFRunLoopRun + 1371
4   com.apple.CoreFoundation   0x00007fff9a8c0bd8 CFRunLoopRunSpecific + 296
…
10  com.apple.AppKit           0x00007fff8e823c03 -[NSApplication run] + 594
11  com.apple.AppKit           0x00007fff8e7a0354 NSApplicationMain + 1832
12  com.example                0x0000000100013b4 start + 52
```

Reading the Tea Leaves

Main queue

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
                                                  <my code>
12  com.apple.Foundation      0x00007fff931157e8 __NSBLOCKOPERATION_IS_CALLING_OUT_TO_A_BLOCK__ + 7
13  com.apple.Foundation      0x00007fff931155b5 -[NSBlockOperation main] + 9
14  com.apple.Foundation      0x00007fff93114a6c -[__NSOperationInternal _start:] + 653
15  com.apple.Foundation      0x00007fff93114543 __NSOQSchedule_f + 184
16  libdispatch.dylib         0x00007fff935d6c13 _dispatch_client_callout + 8
17  libdispatch.dylib         0x00007fff935e2cbf _dispatch_main_queue_callback_4CF + 861
18  com.apple.CoreFoundation  0x00007fff8d9223f9 __CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__
19  com.apple.CoreFoundation  0x00007fff8d8dd68f __CFRunLoopRun + 2159
20  com.apple.CoreFoundation  0x00007fff8d8dcbd8 CFRunLoopRunSpecific + 296
…
26  com.apple.AppKit          0x00007fff999a1bd3 -[NSApplication run] + 594
27  com.apple.AppKit          0x00007fff9991e324 NSApplicationMain + 1832
28  libdyld.dylib             0x00007fff9480f5c9 start + 1
```

Summary

An efficient and responsive application must be able to adopt to diverse environments

Summary

An efficient and responsive application must be able to adopt to diverse environments

QoS classes enable the system to manage resources appropriately

Summary

An efficient and responsive application must be able to adopt to diverse environments

QoS classes enable the system to manage resources appropriately

Integrate QoS into your application and existing use of GCD

Summary

An efficient and responsive application must be able to adopt to diverse environments

QoS classes enable the system to manage resources appropriately

Integrate QoS into your application and existing use of GCD

Consider your app's use of GCD and avoid thread explosion

More Information

Documentation

Concurrency Programming Guide

https://developer.apple.com/library/ios/documentation/General/Conceptual/ConcurrencyProgrammingGuide/

Energy Efficiency Guide for Mac Apps

https://developer.apple.com/library/mac/documentation/Performance/Conceptual/power_efficiency_guidelines_osx/

Energy Efficiency Guide for iOS Apps

https://developer.apple.com/library/prerelease/ios/documentation/Performance/Conceptual/EnergyGuide-iOS/

More Information

Technical Support

Apple Developer Forums

http://developer.apple.com/forums

Developer Technical Support

http://developer.apple.com/support/technical

General Inquiries

Paul Danbold, Core OS Evangelist

danbold@apple.com

Related Sessions

Achieving All-Day Battery Life	Nob Hill	Wednesday 9:00AM
Optimizing Your App for Multitasking on iPad in iOS 9	Presidio	Wednesday 3:30PM
Advanced NSOperations	Presidio	Friday 9:00AM
Performance on iOS and watchOS	Presidio	Friday 11:00AM

Related Labs

Power and Performance Lab	Frameworks Lab C	Friday 12:00PM

WWDC15