Building Concurrent User Interfaces on iOS

Session 211

Andy Matuschak
iOS Frameworks

Our Roadmap

Our Roadmap

- Concurrent processing of data intended for UI

Our Roadmap

- Concurrent processing of data intended for UI
- Concurrent drawing of UI graphics

Our Roadmap

- Concurrent processing of data intended for UI
- Concurrent drawing of UI graphics
- Canceling concurrent operations


3G
9:41 AM
Categories
Mid-Yield
High-Risk
Undervalued


3G
9:41 AM
Categories
Mid-Cap
SQXY
TXNJ
LEEB
KPAI
JJPP

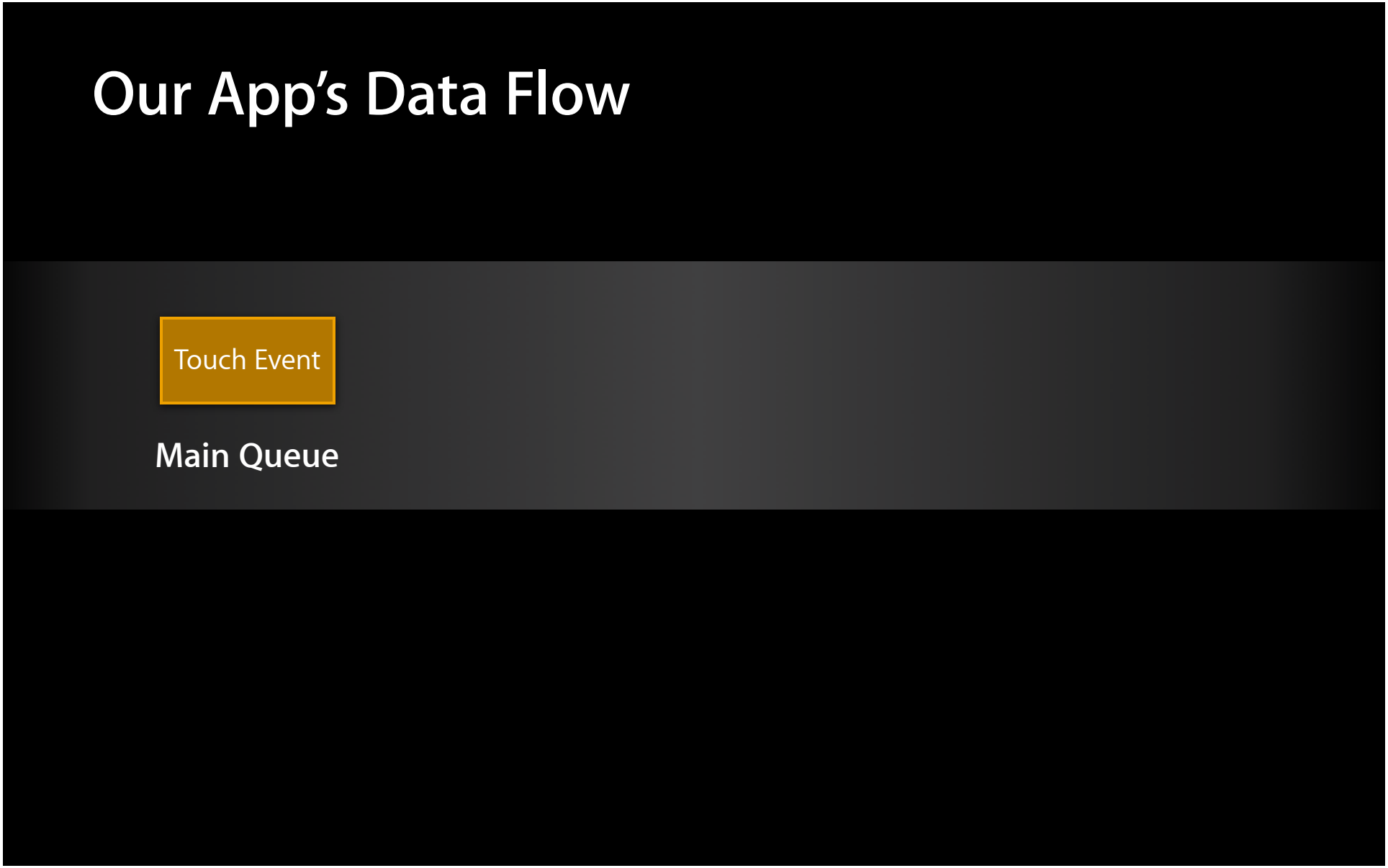
Demo

Our App's Data Flow

Our App's Data Flow

Main Queue

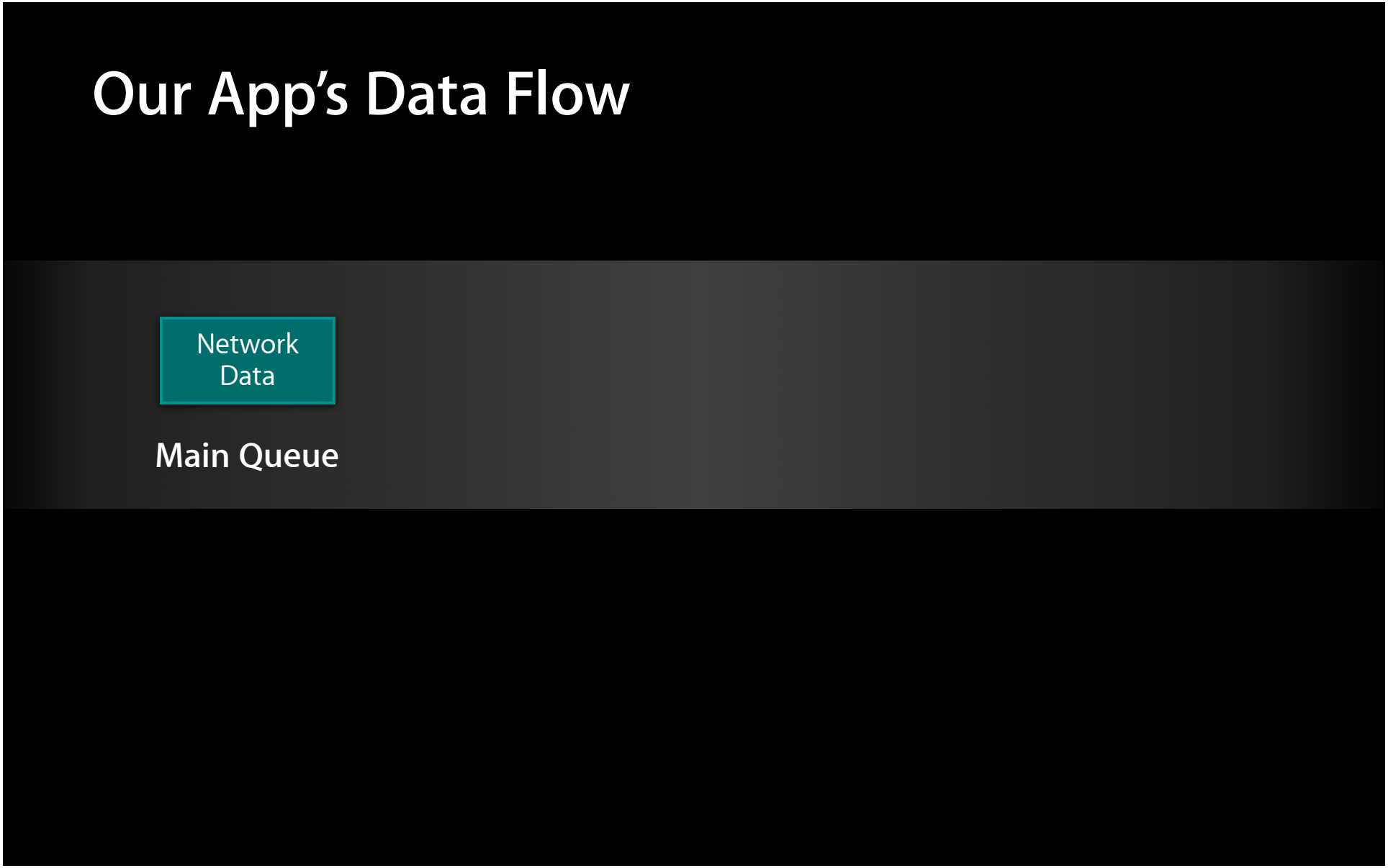
Our App's Data Flow



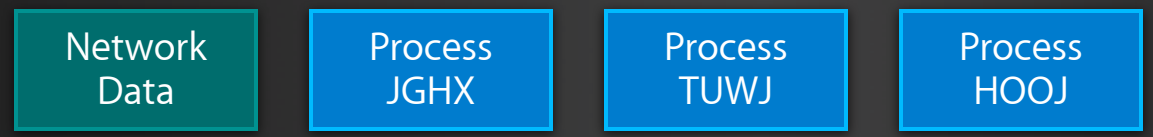
Our App's Data Flow

Main Queue

Our App's Data Flow

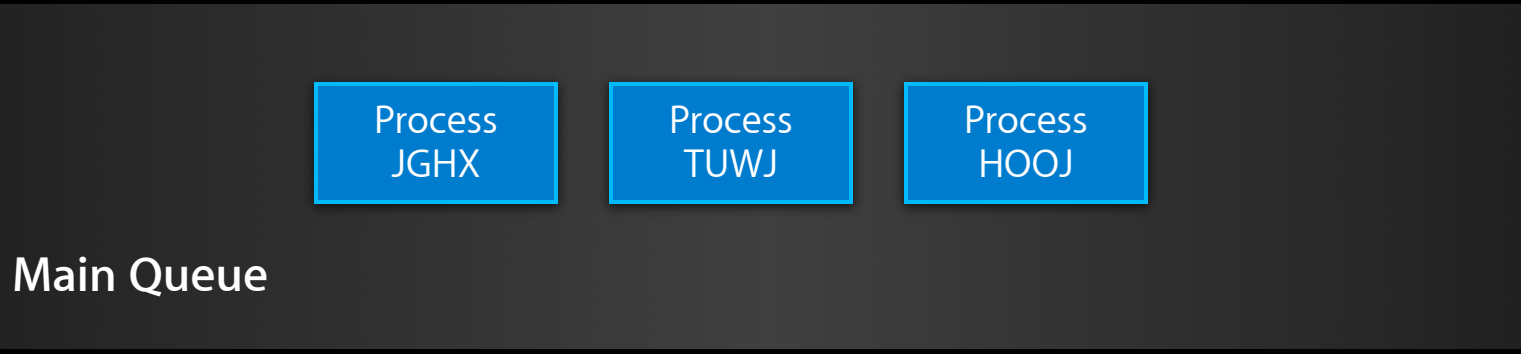


Our App's Data Flow



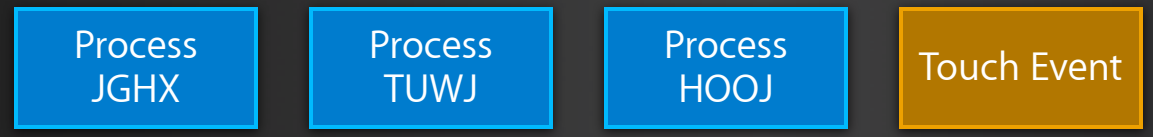
Main Queue

Our App's Data Flow



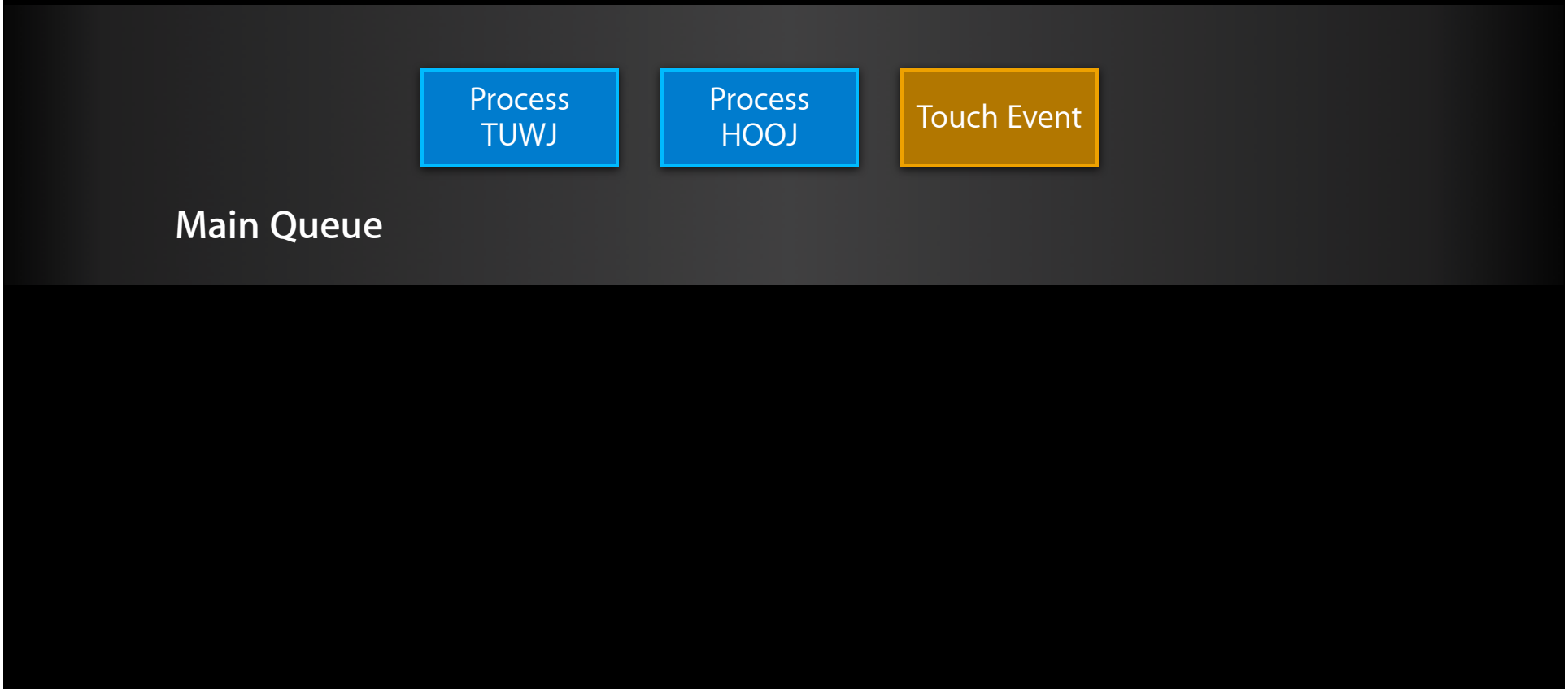
Our App's Data Flow

Our App's Data Flow

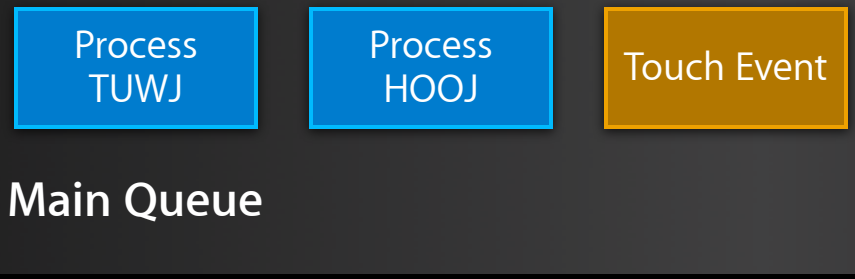


Main Queue

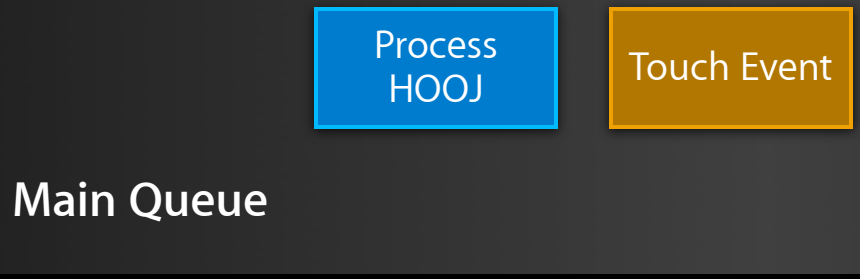
Our App's Data Flow



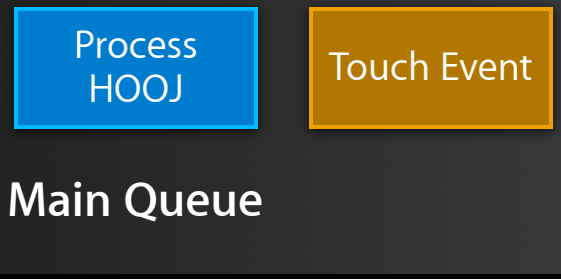
Our App's Data Flow



Our App's Data Flow

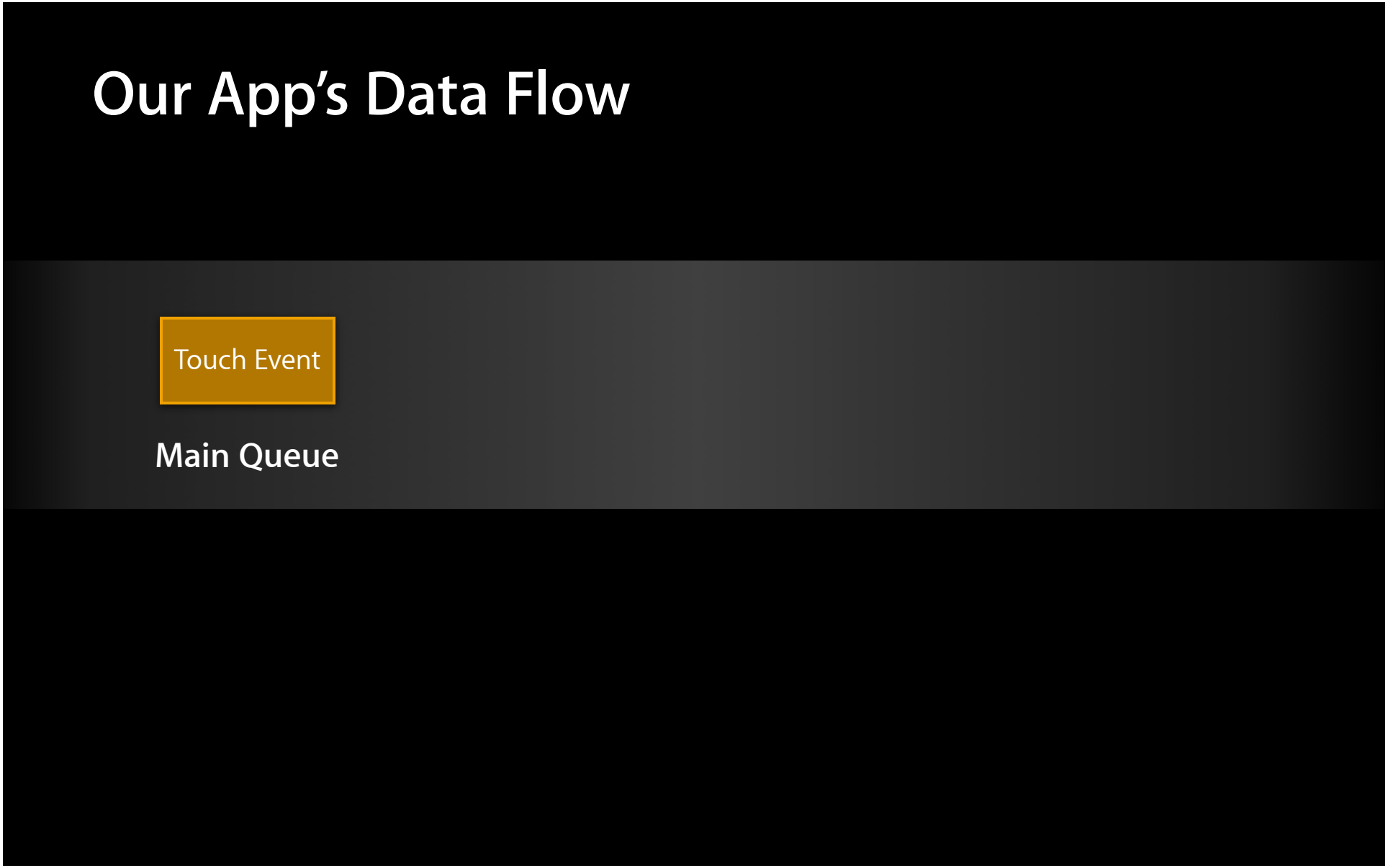


Our App's Data Flow



Our App's Data Flow

Our App's Data Flow



Our App's Data Flow

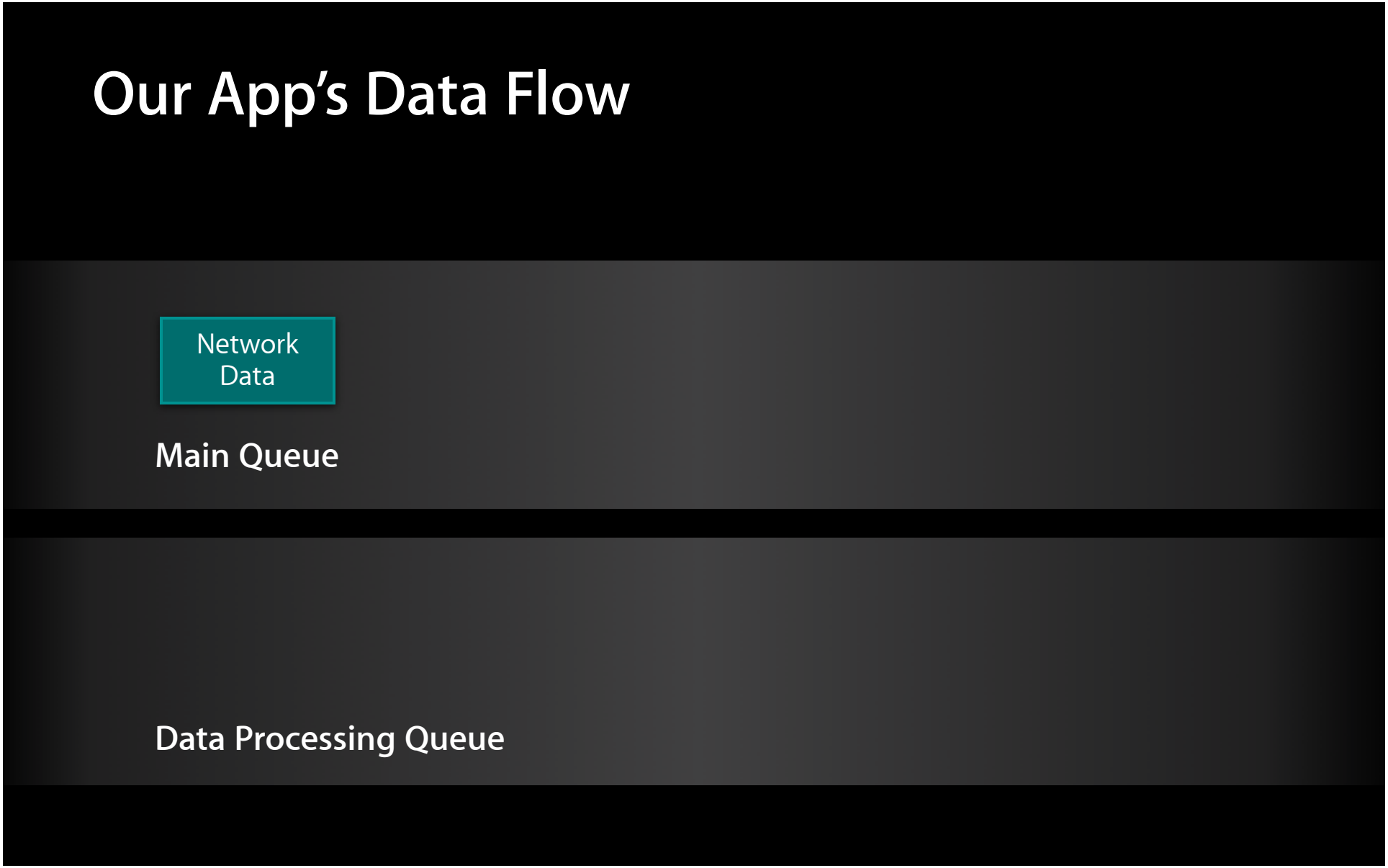
Main Queue

Our App's Data Flow

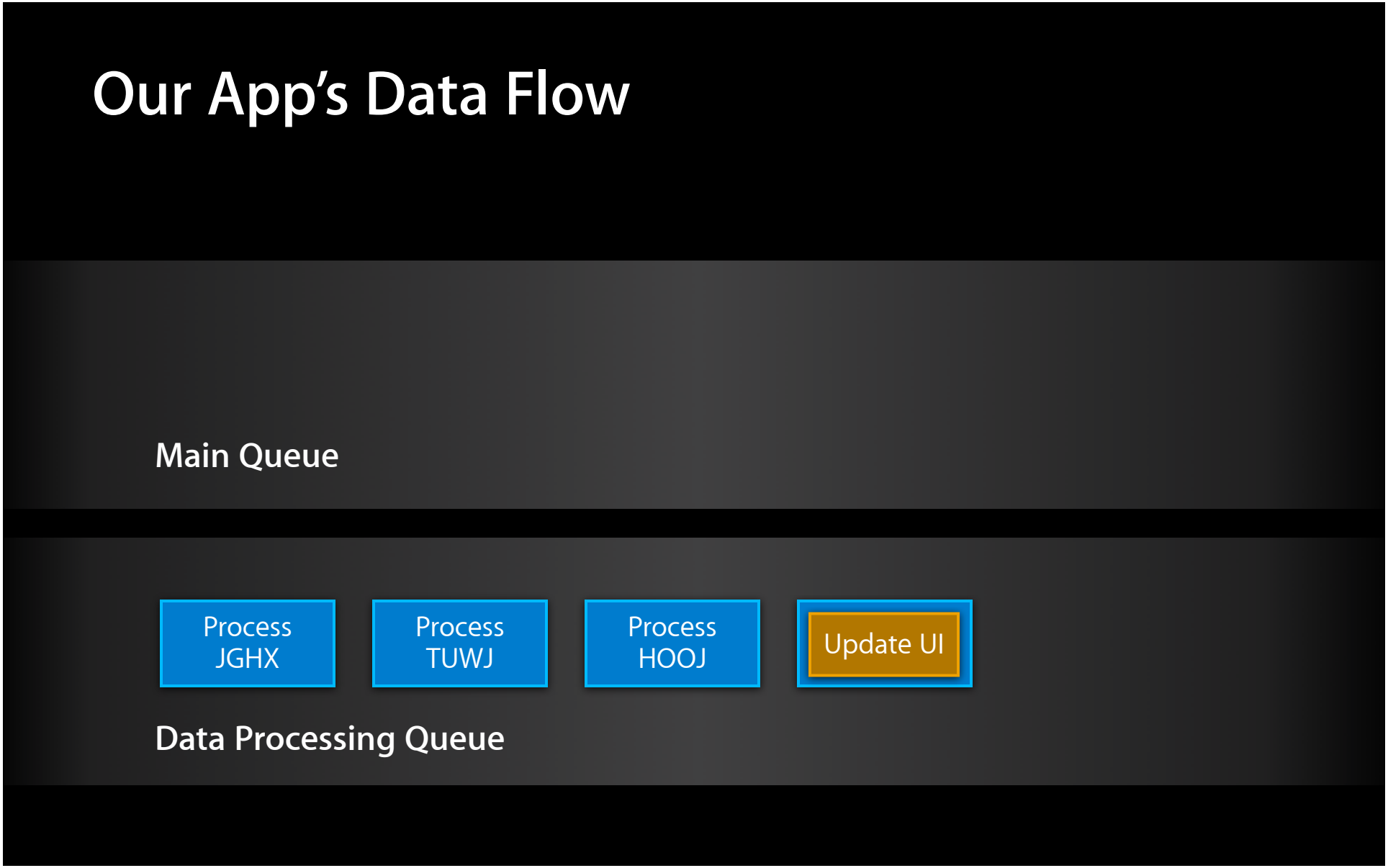
Main Queue

Data Processing Queue

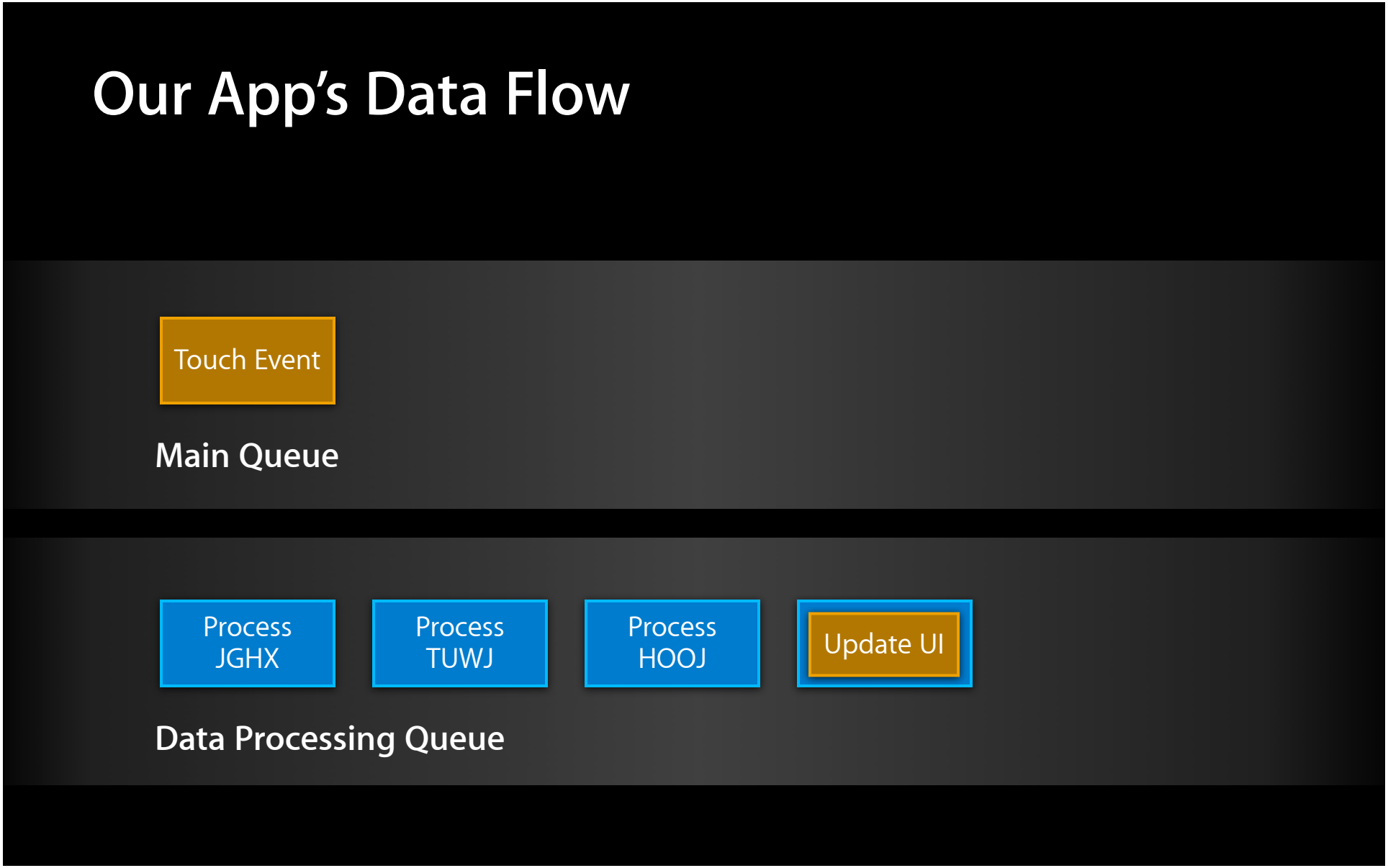
Our App's Data Flow



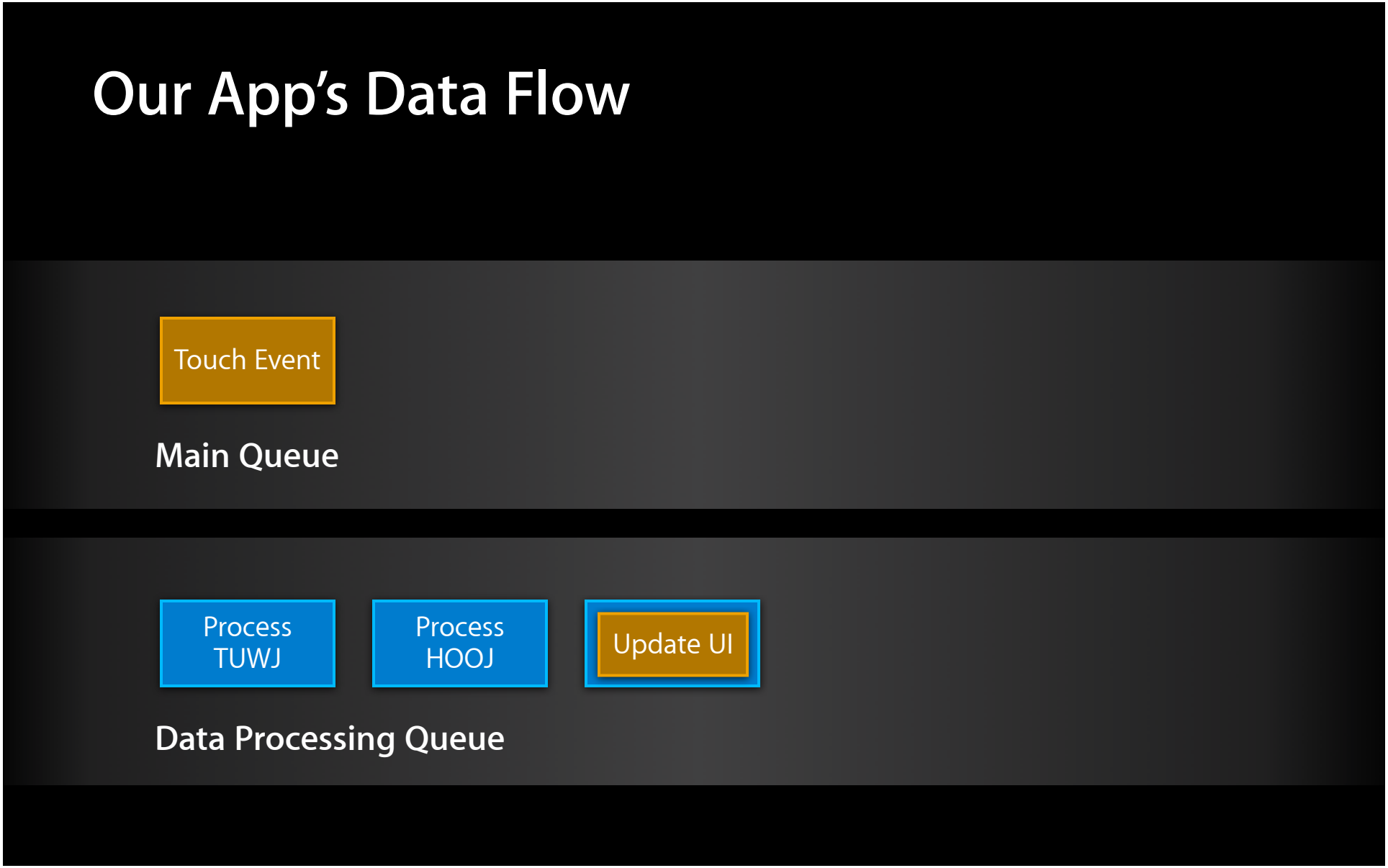
Our App's Data Flow



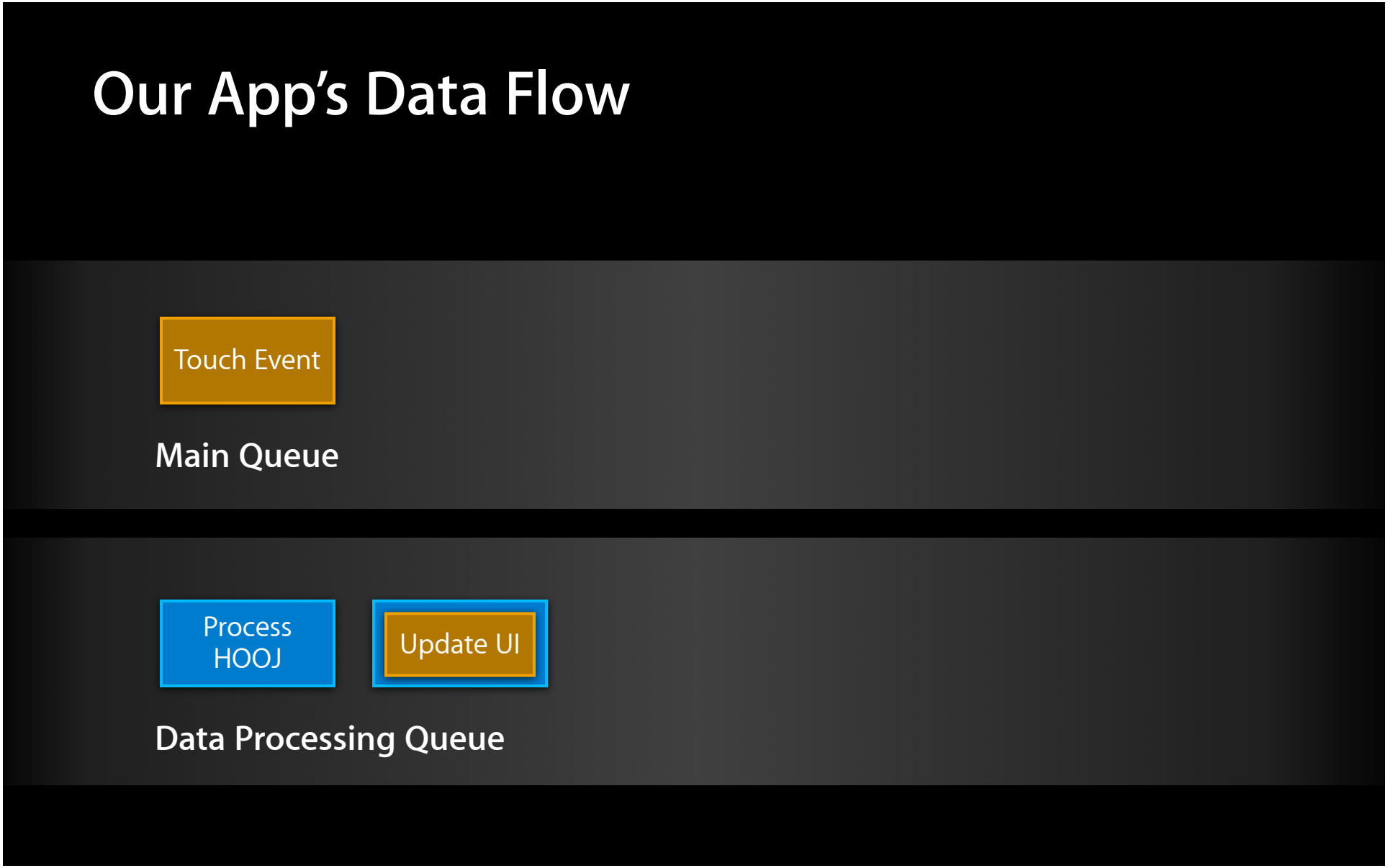
Our App's Data Flow



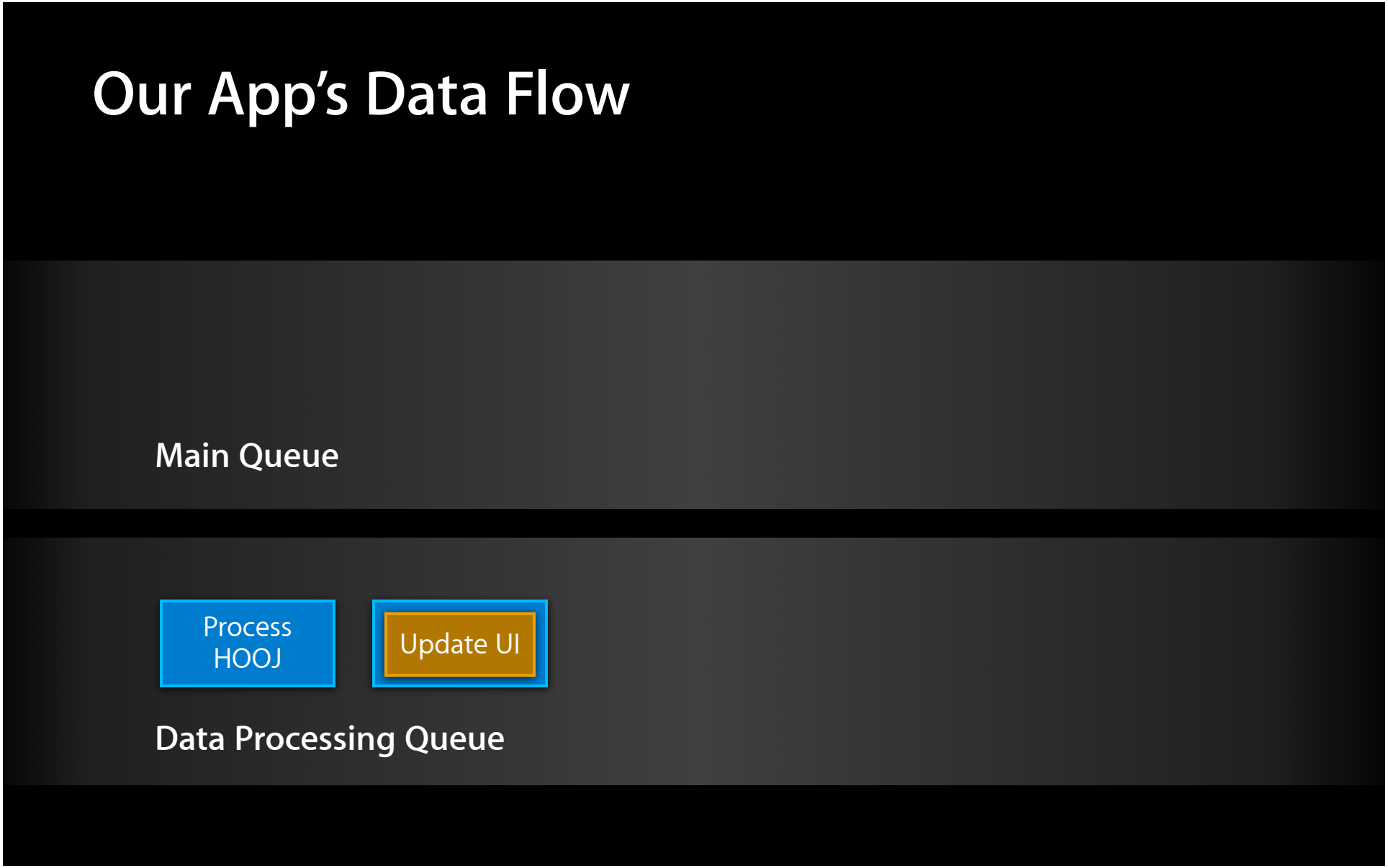
Our App's Data Flow



Our App's Data Flow



Our App's Data Flow



Our App's Data Flow

Main Queue

Data Processing Queue

Our App's Data Flow

Main Queue

Data Processing Queue

Our App's Data Flow

Update UI

Main Queue

Data Processing Queue

Making Operation Queues

Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
```

Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@”Data Processing Queue”];
```

Data Processing Queue

Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@”Data Processing Queue”];
[queue addOperationWithBlock:^{ processStock(someStock); }];
```

Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@"Data Processing Queue"];
[queue addOperationWithBlock:^{ processStock(someStock); }];
[queue addOperationWithBlock:^{
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        updateUI(someStock);
    }];
}];
```

Data Processing Queue

Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@"Data Processing Queue"];
[queue addOperationWithBlock:^{ processStock(someStock); }];
[queue addOperationWithBlock:^{
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        updateUI(someStock);
    }];
}];
```

Data Processing Queue

Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@"Data Processing Queue"];
[queue addOperationWithBlock:^{ processStock(someStock); }];
[queue addOperationWithBlock:^{
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        updateUI(someStock);
    }];
}];
```

Data Processing Queue

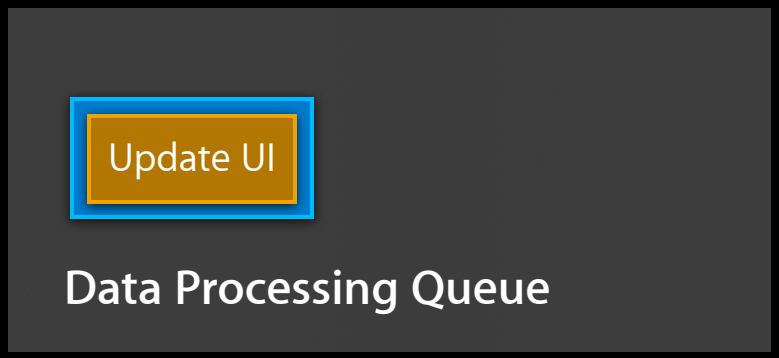
Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@"Data Processing Queue"];
[queue addOperationWithBlock:^{ processStock(someStock); }];
[queue addOperationWithBlock:^{
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        updateUI(someStock);
    }];
}];
```

Data Processing Queue

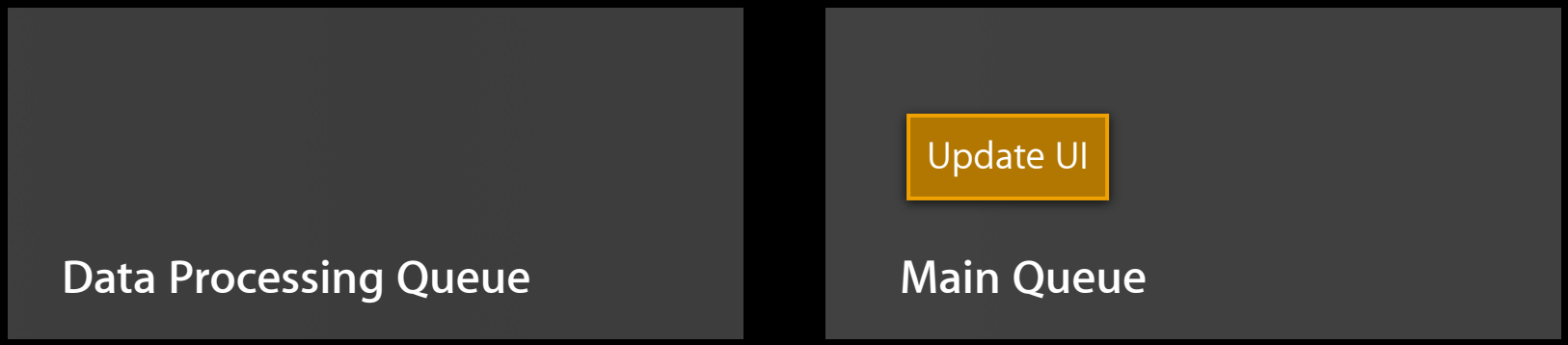
Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@"Data Processing Queue"];
[queue addOperationWithBlock:^{ processStock(someStock); }];
[queue addOperationWithBlock:^{
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        updateUI(someStock);
    }];
}];
```



Making Operation Queues

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue setName:@"Data Processing Queue"];
[queue addOperationWithBlock:^{ processStock(someStock); }];
[queue addOperationWithBlock:^{
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        updateUI(someStock);
    }];
}];
```



Demo

Concurrent data processing

Concurrent Data Processing

Review

Concurrent Data Processing

Review

- System events on the main queue

Concurrent Data Processing

Review

- System events on the main queue
- Separate expensive processing with `NSOperationQueue`

Concurrent Data Processing

Review

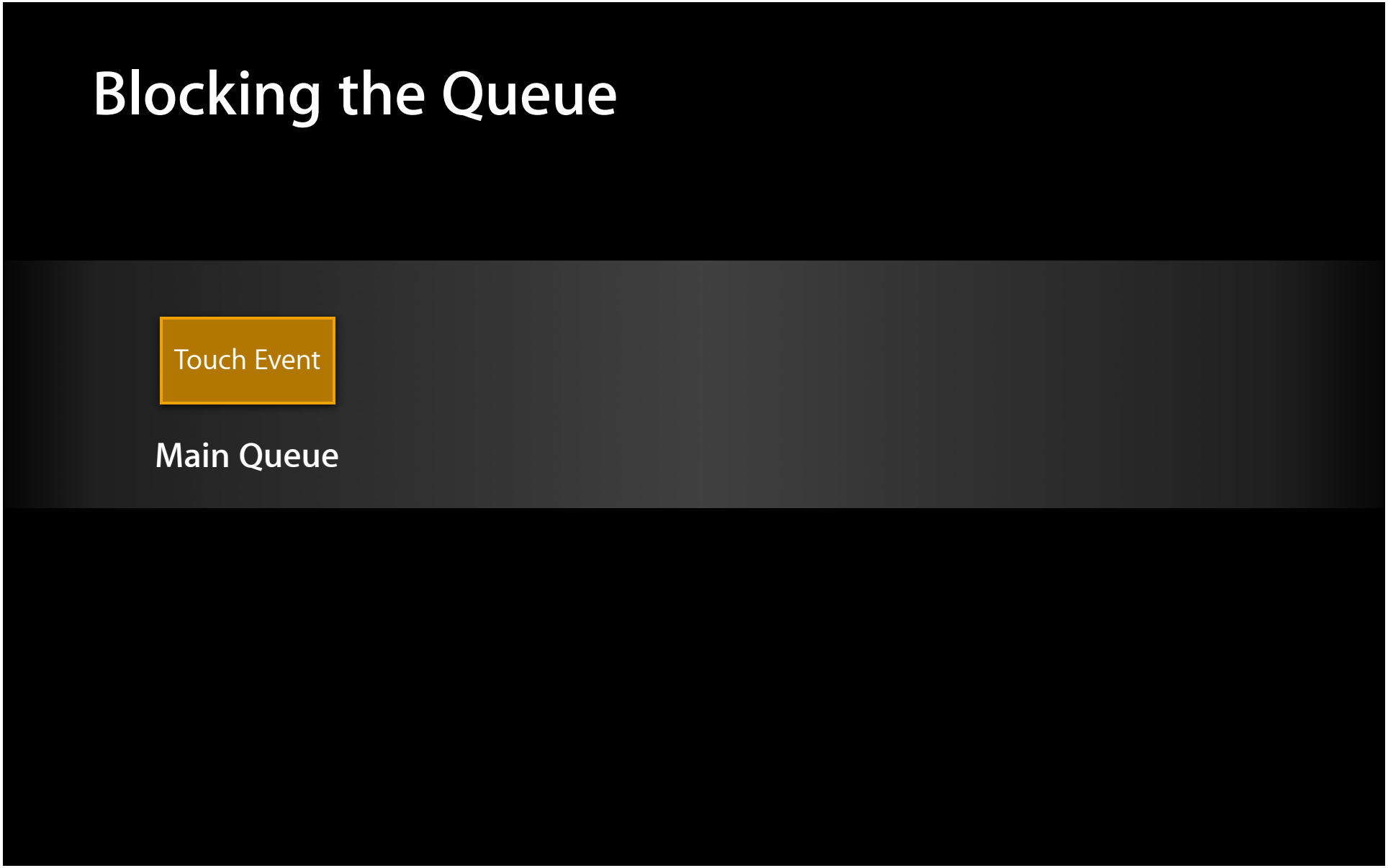
- System events on the main queue
- Separate expensive processing with `NSOperationQueue`
- Update data UIKit accesses on the main queue

Concurrent Drawing

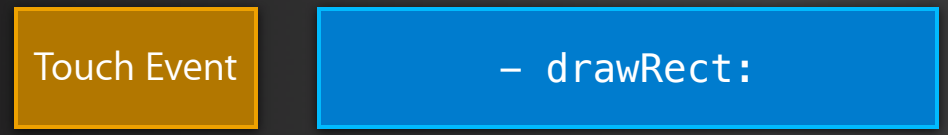
Blocking the Queue

Main Queue

Blocking the Queue



Blocking the Queue



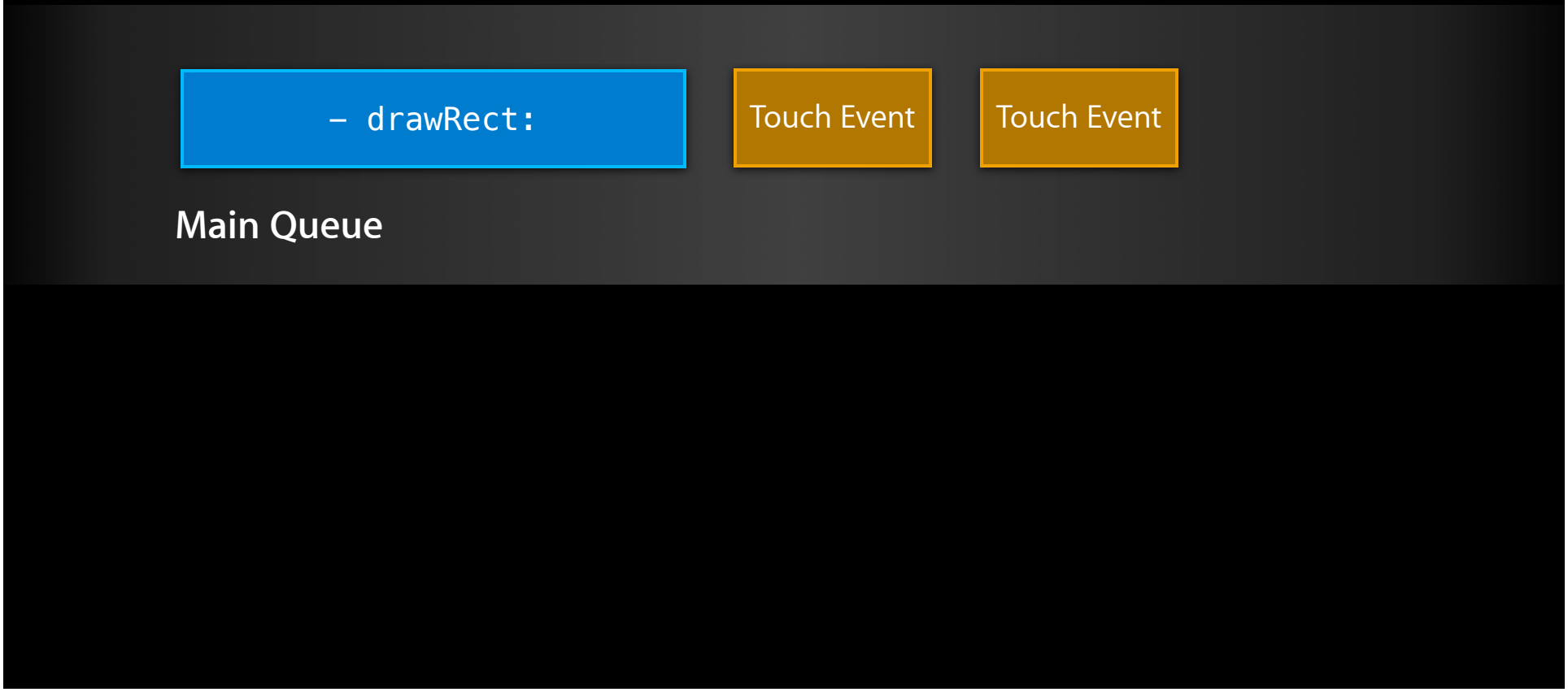
Main Queue

Blocking the Queue

```
- drawRect:
```

Main Queue

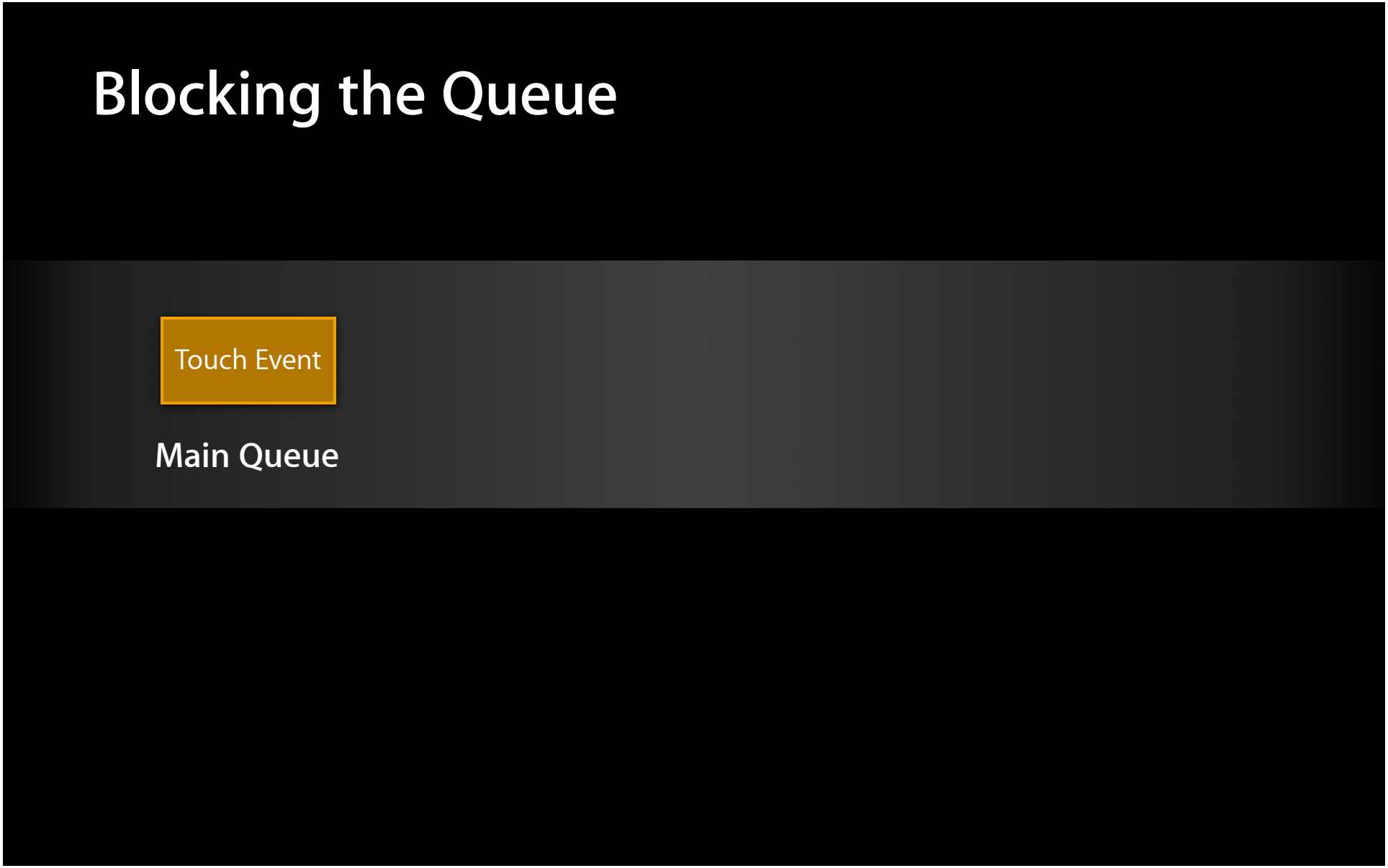
Blocking the Queue



Blocking the Queue

Main Queue

Blocking the Queue

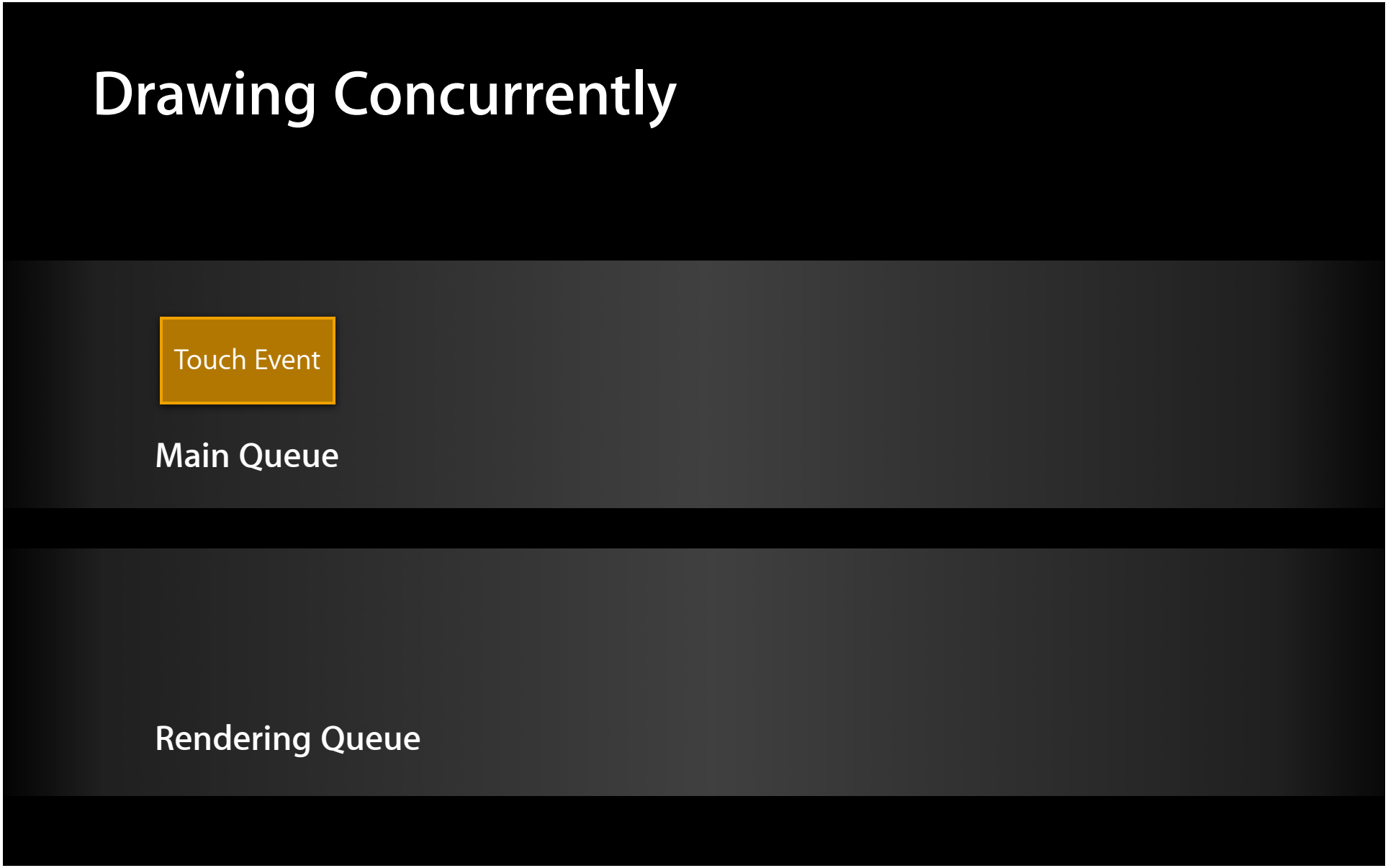


Drawing Concurrently

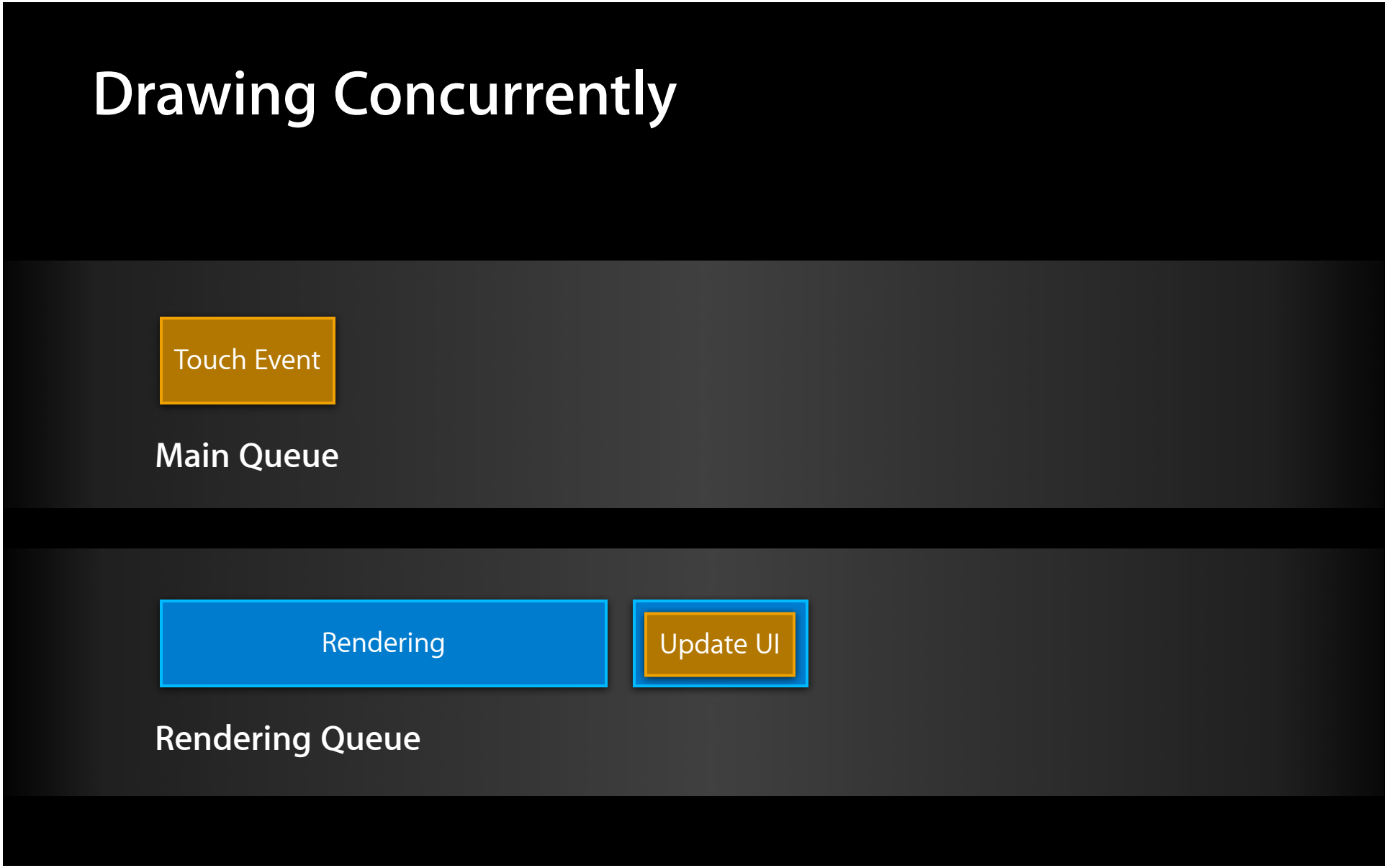
Main Queue

Rendering Queue

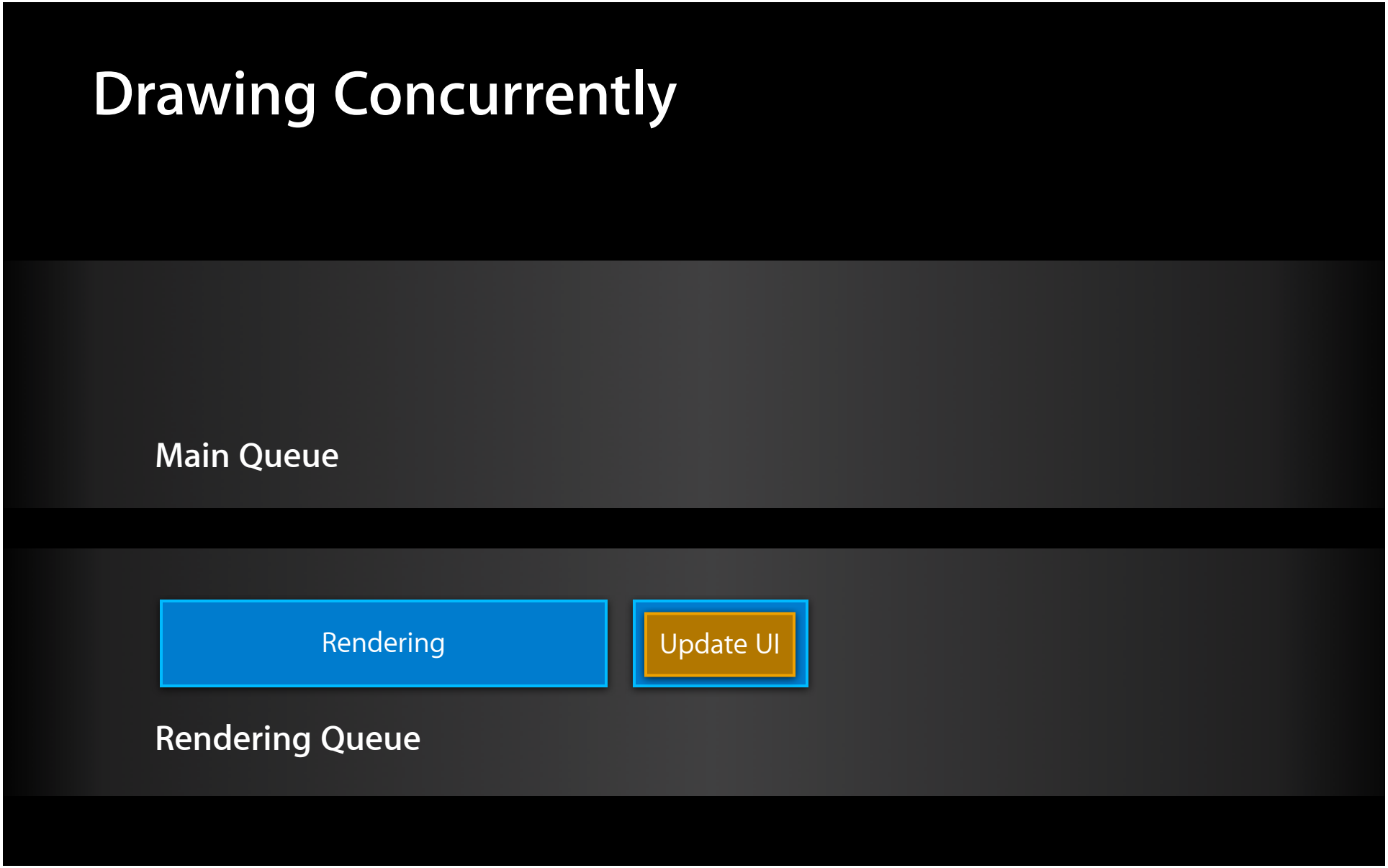
Drawing Concurrently



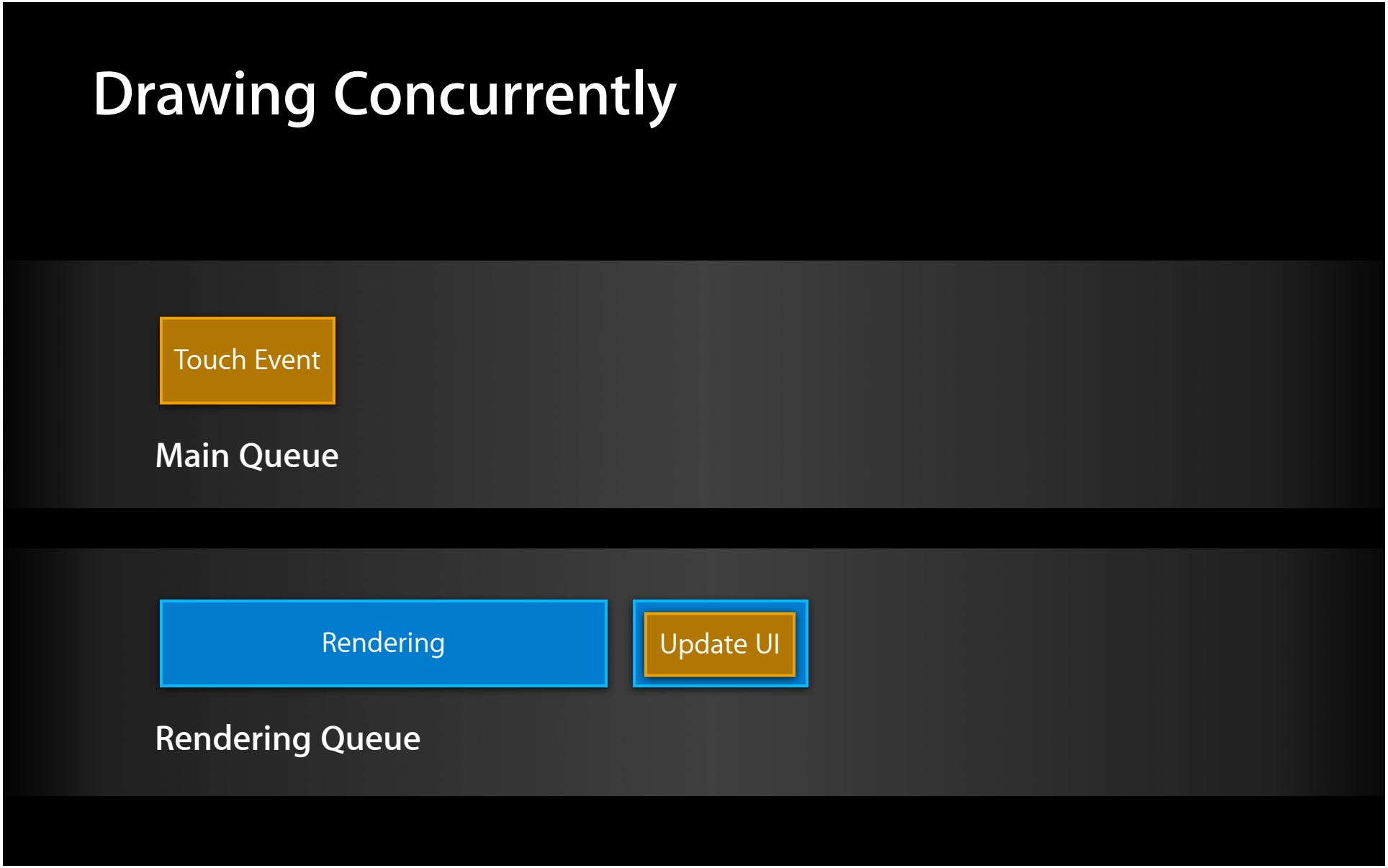
Drawing Concurrently



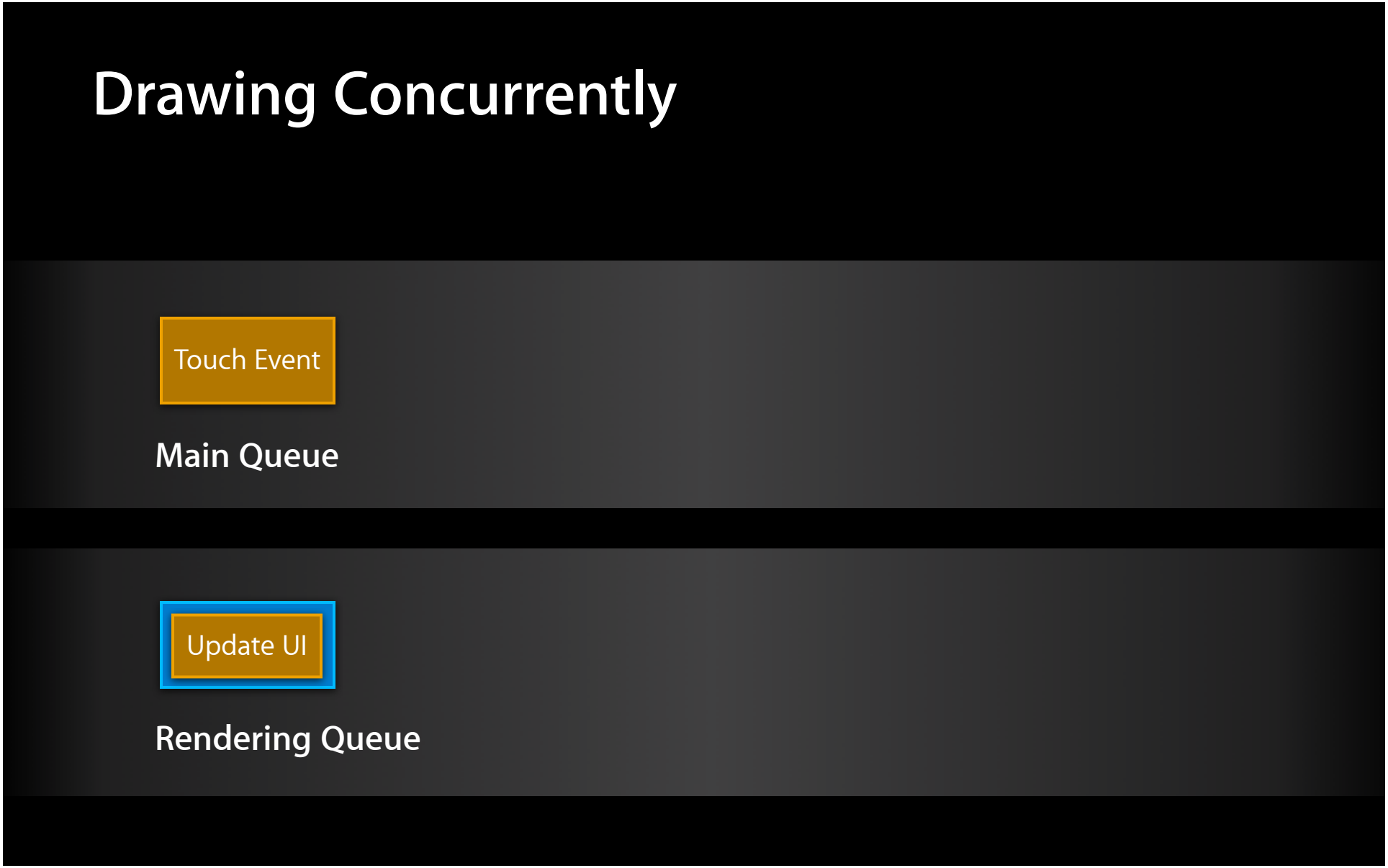
Drawing Concurrently



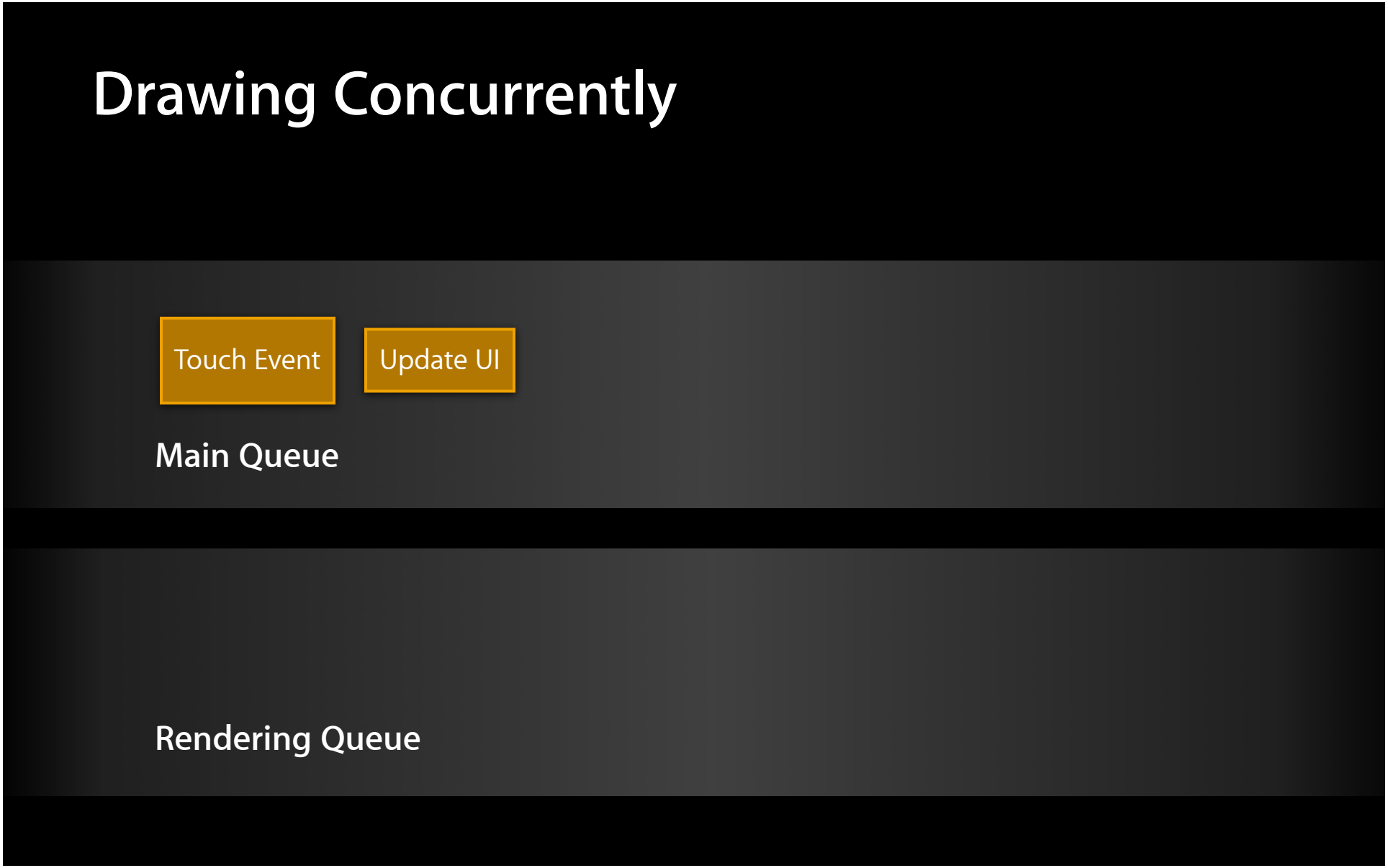
Drawing Concurrently



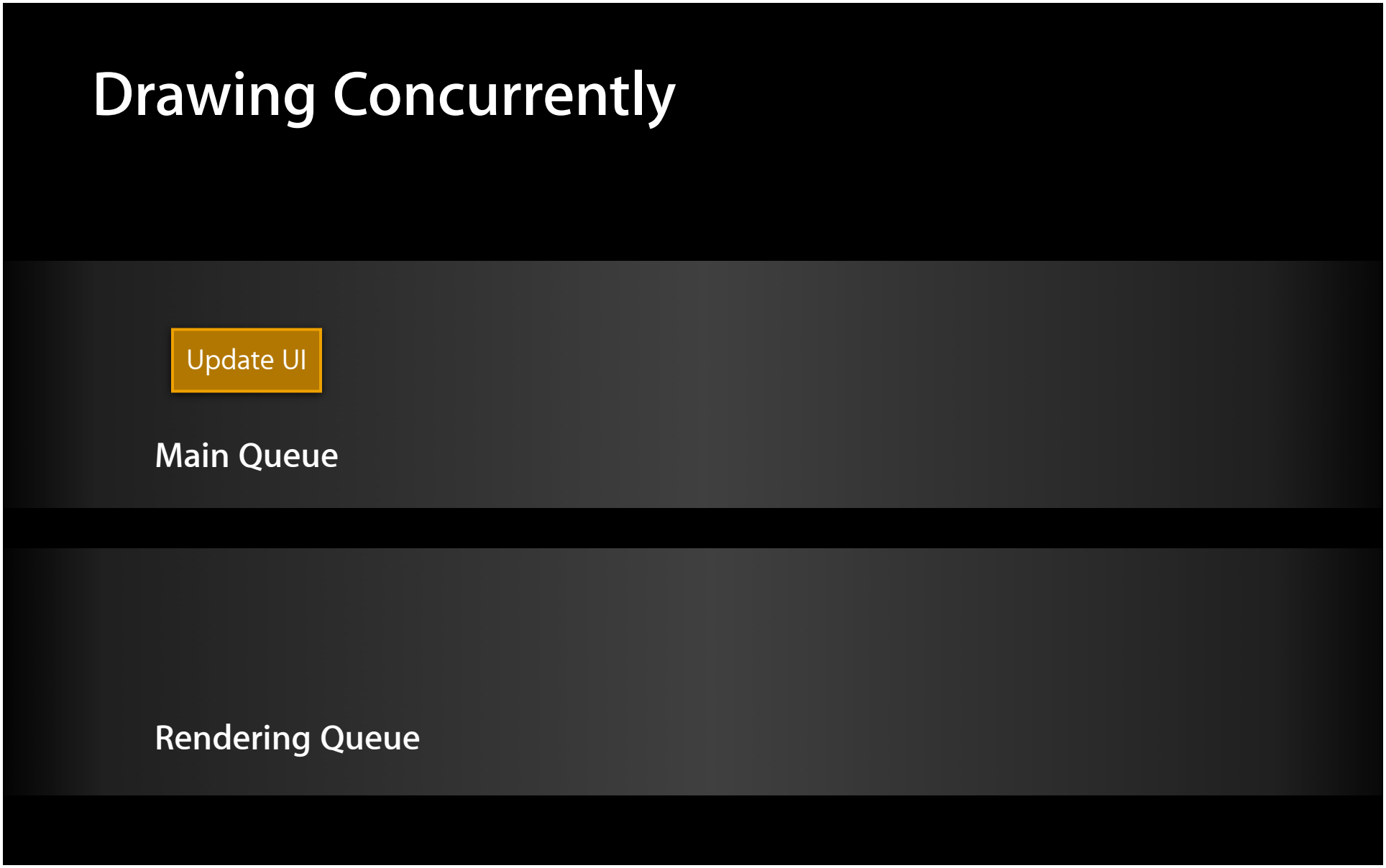
Drawing Concurrently



Drawing Concurrently



Drawing Concurrently

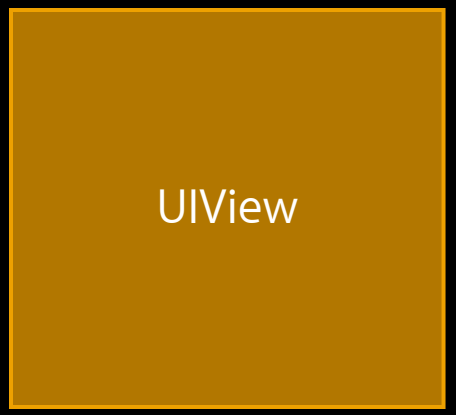


Drawing Concurrently

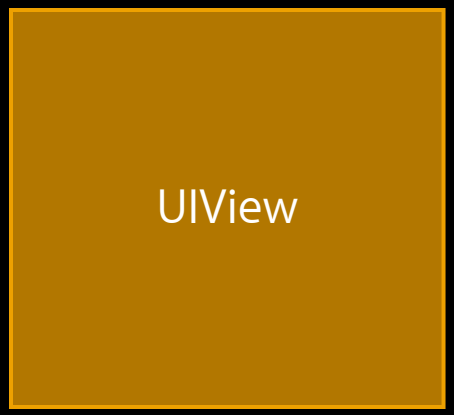
Main Queue

Rendering Queue

UIView Drawing Model

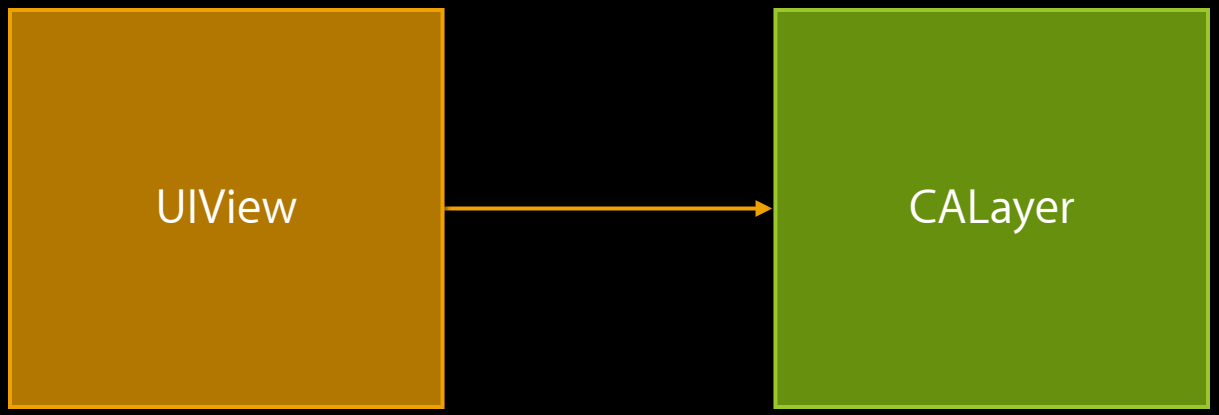


UIView Drawing Model



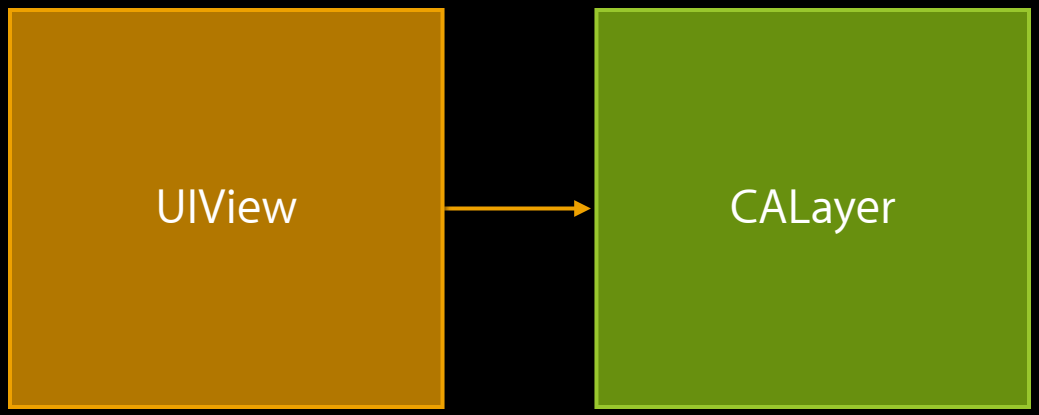

```
[view setNeedsDisplay]
```

UIView Drawing Model




```
[view setNeedsDisplay]   [layer setNeedsDisplay]
```

UIView Drawing Model

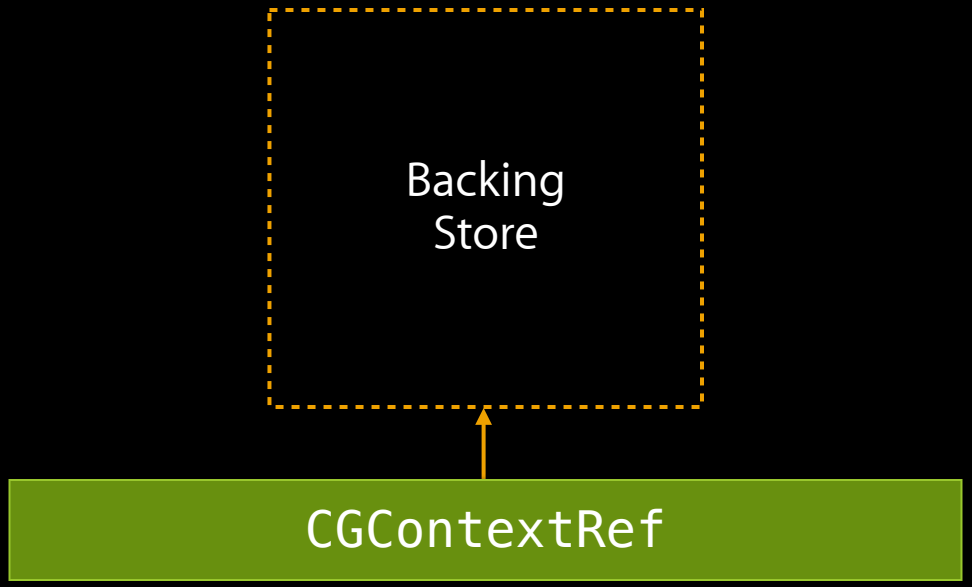


UIView Drawing Model

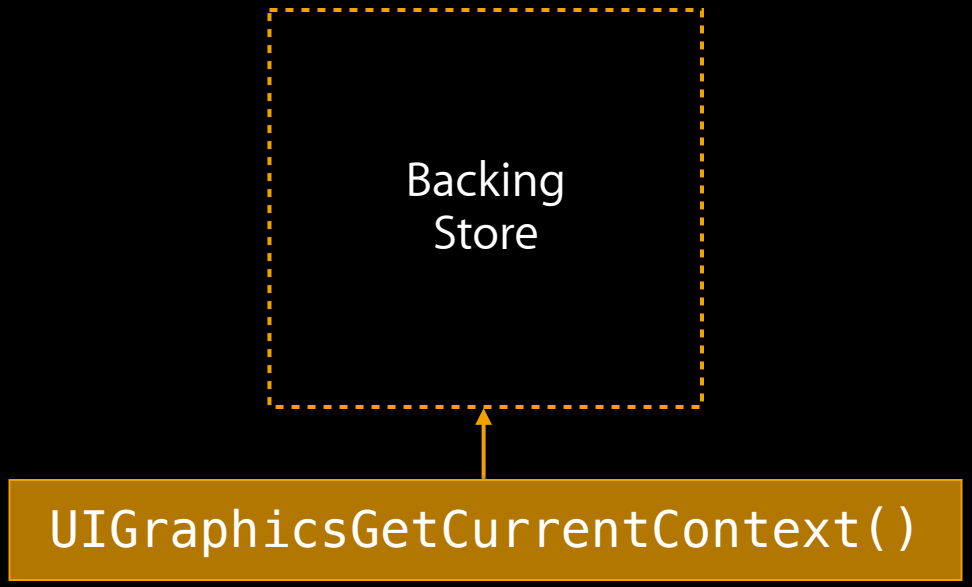

```
[layer display]
```

UIView Drawing Model

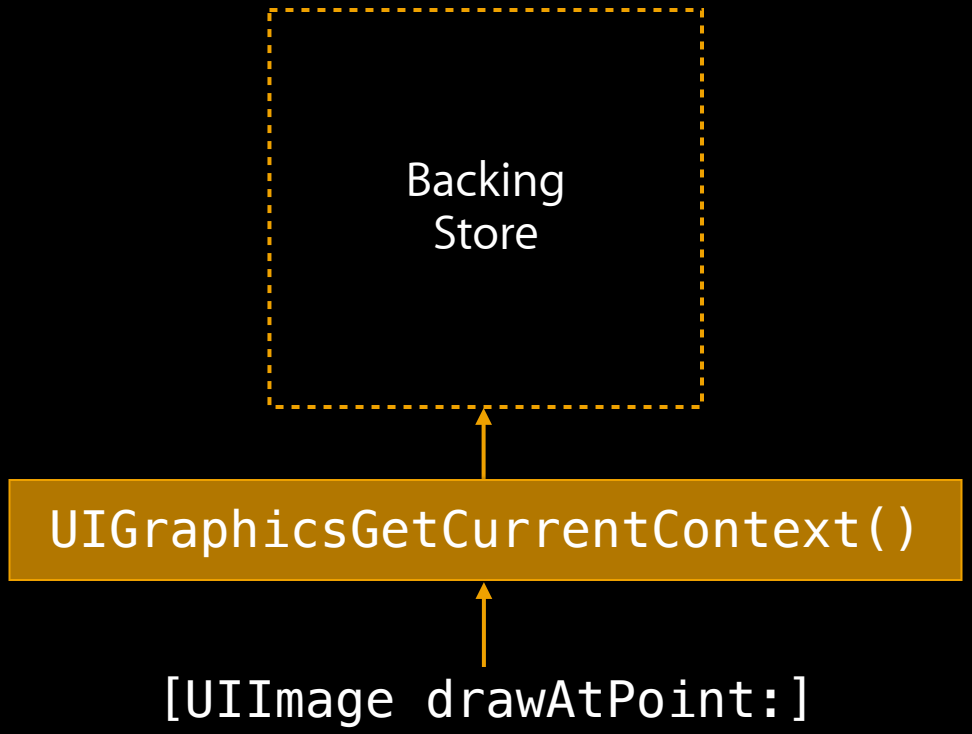
UIView Drawing Model



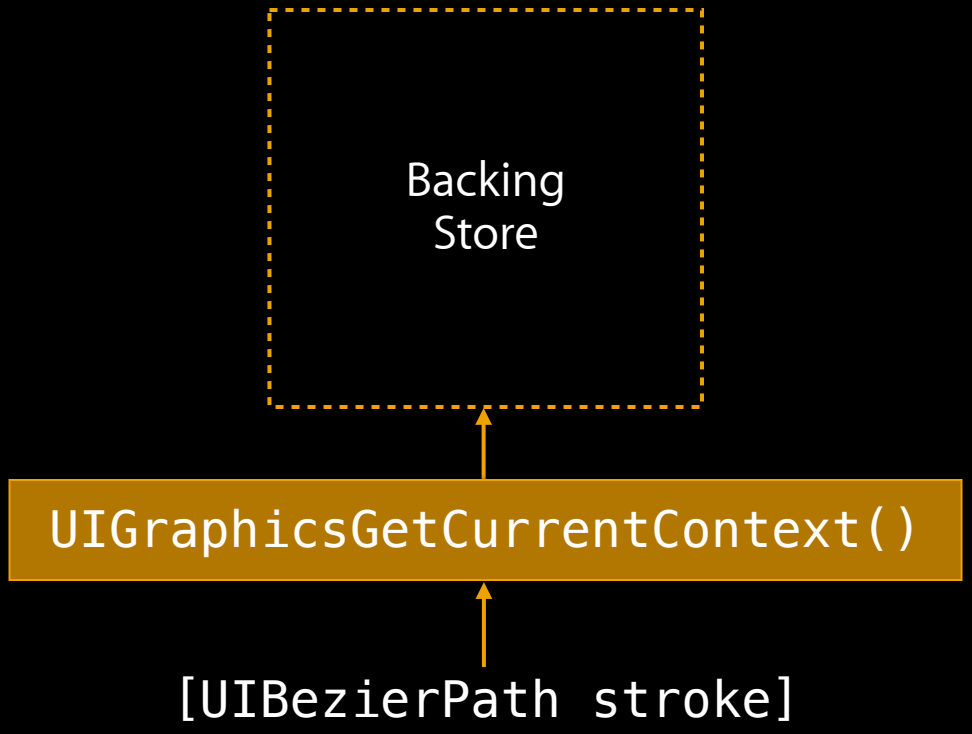
UIView Drawing Model



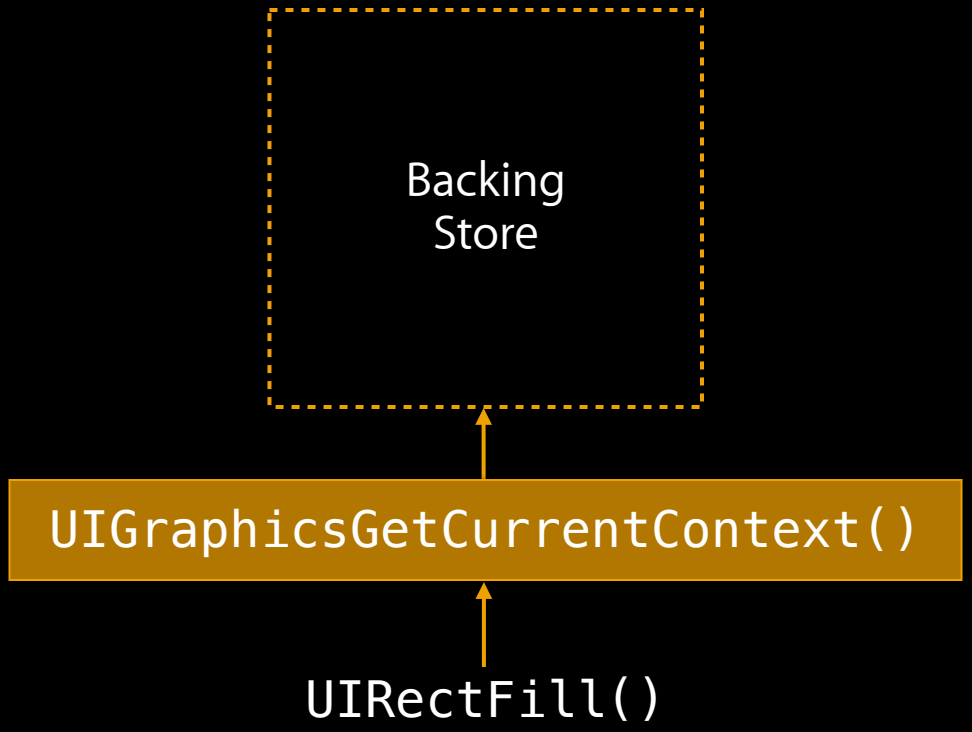
UIView Drawing Model



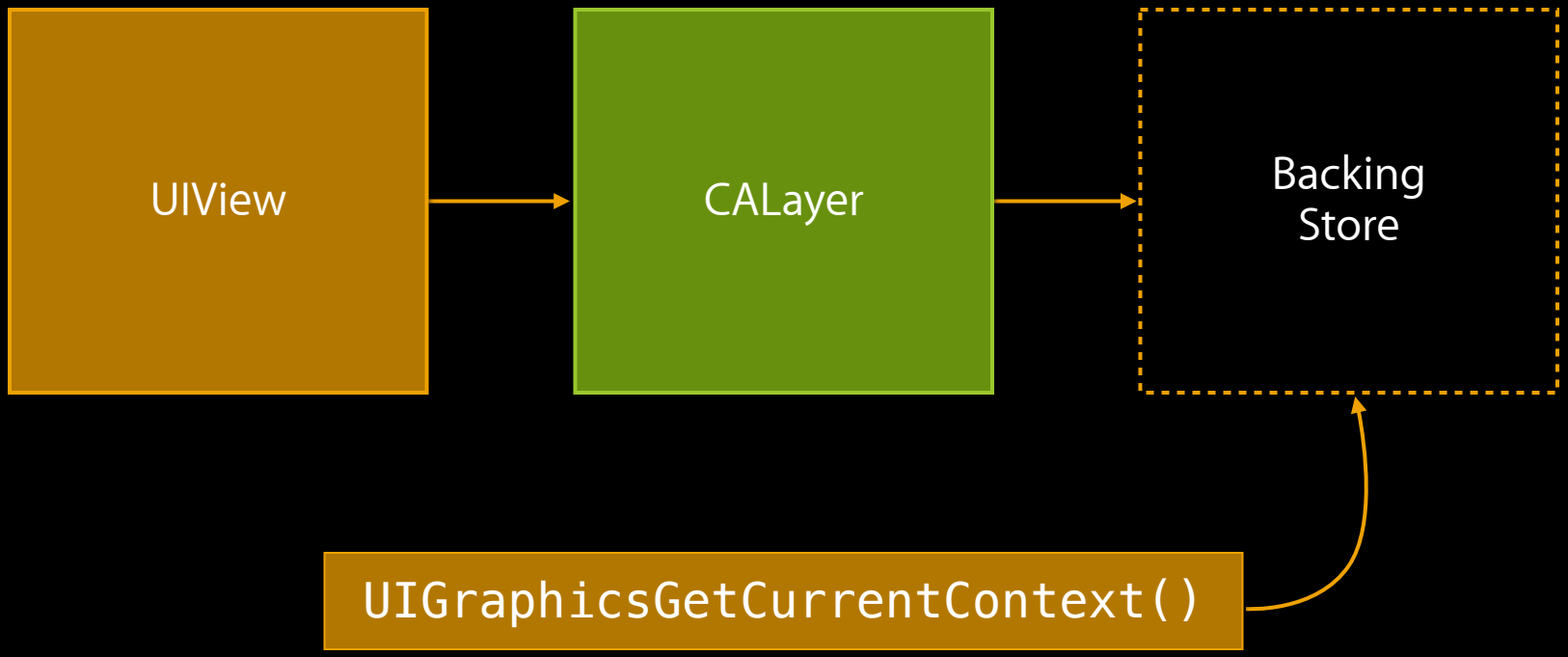
UIView Drawing Model



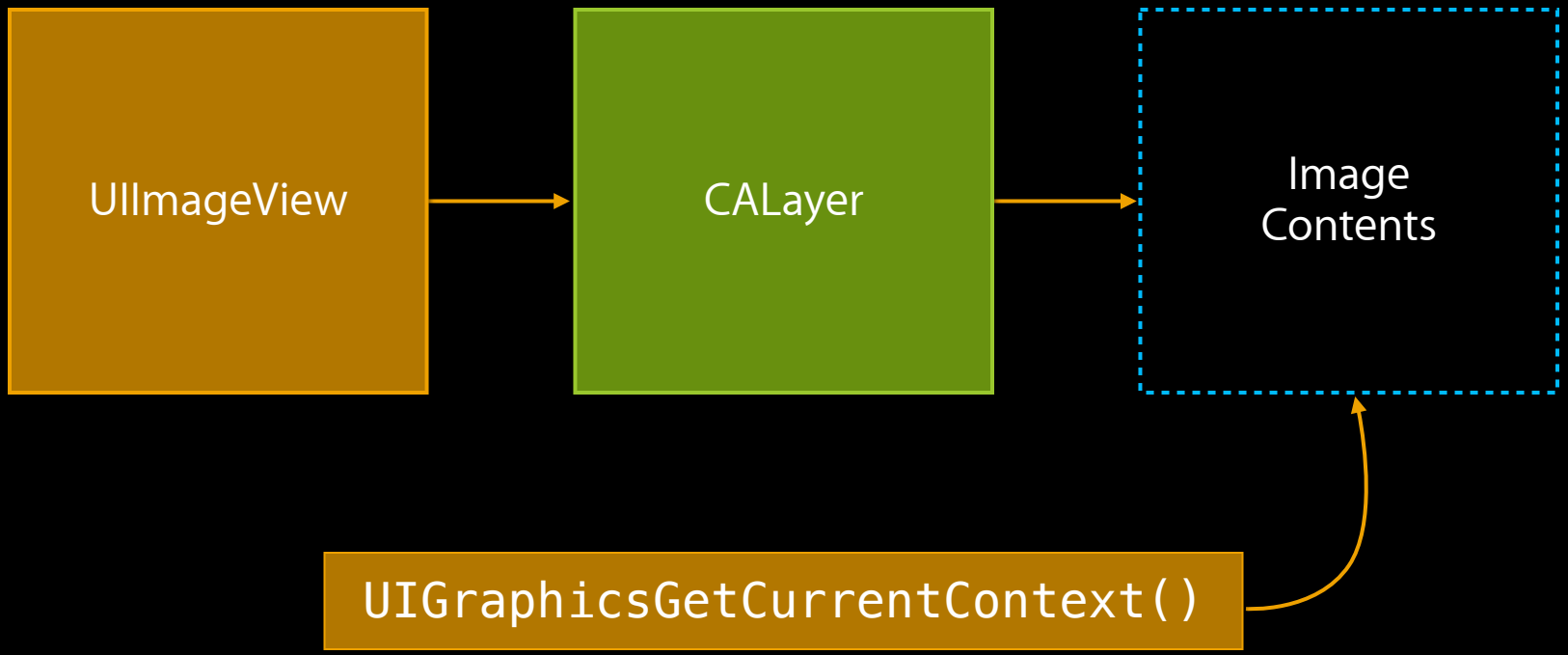
UIView Drawing Model



Drawing Concurrently



Drawing Concurrently



Adapting drawRect:

```
- (void)drawRect:(CGRect)rect {
    [[UIColor greenColor] set];
    UIRectFill([self bounds]);
    [anImage drawAtPoint:CGPointZero];
}
```

Adapting drawRect:

```
[[UIColor greenColor] set];
UIRectFill([self bounds]);
[anImage drawAtPoint:CGPointZero];
```

Adapting drawRect:

```
- (UIImage *)renderInImageOfSize:(CGSize)size {


    [[UIColor greenColor] set];
    UIRectFill([self bounds]);
    [anImage drawAtPoint:CGPointZero];




}
```

Adapting drawRect:

```
- (UIImage *)renderInImageOfSize:(CGSize)size {

    UIGraphicsBeginImageContextWithOptions(size, NO, 0);

    [[UIColor greenColor] set];
    UIRectFill([self bounds]);
    [anImage drawAtPoint:CGPointZero];




}
```

Adapting drawRect:

```
- (UIImage *)renderInImageOfSize:(CGSize)size {

    UIGraphicsBeginImageContextWithOptions(size, NO, 0);

    [[UIColor greenColor] set];
    UIRectFill([self bounds]);
    [anImage drawAtPoint:CGPointZero];

    UIImage *i = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return i;

}
```

Demo

Concurrent drawing

Concurrent Drawing

Review

Concurrent Drawing

Review

- UIKit calls `-drawRect:` on the main queue

Concurrent Drawing

Review

- UIKit calls `-drawRect:` on the main queue
- You can draw an image for your view on another queue

Concurrent Drawing

Review

- UIKit calls `-drawRect:` on the main queue
- You can draw an image for your view on another queue
- Drawing APIs are safe to use from any queue…

Concurrent Drawing

Review

- UIKit calls `-drawRect:` on the main queue
- You can draw an image for your view on another queue
- Drawing APIs are safe to use from any queue…
  - …if they begin and end in the same operation!

Concurrent Drawing

Review

- UIKit calls `-drawRect:` on the main queue
- You can draw an image for your view on another queue
- Drawing APIs are safe to use from any queue…
  - …if they begin and end in the same operation!
- Must call `-[UIImageView setImage:]` on the main queue

Cancellation

Cancellation

Data Processing Queue

Cancellation

Data Processing Queue

```
-[NSOperationQueue cancelAllOperations]
```

Cancellation

Process TUWJ

Data Processing Queue

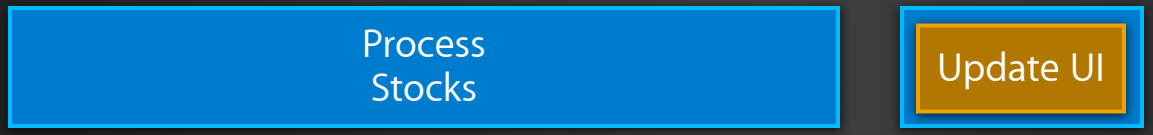
```
-[NSOperationQueue cancelAllOperations]
```

Cancellation

Data Processing Queue

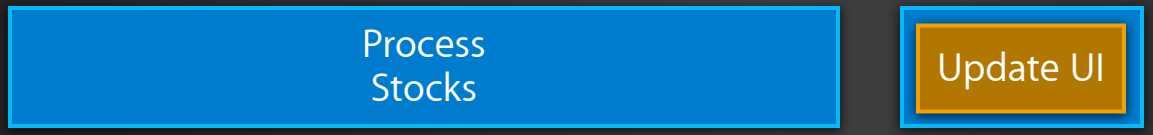
```
-[NSOperationQueue cancelAllOperations]
```

Cancellation



Data Processing Queue

Cancellation



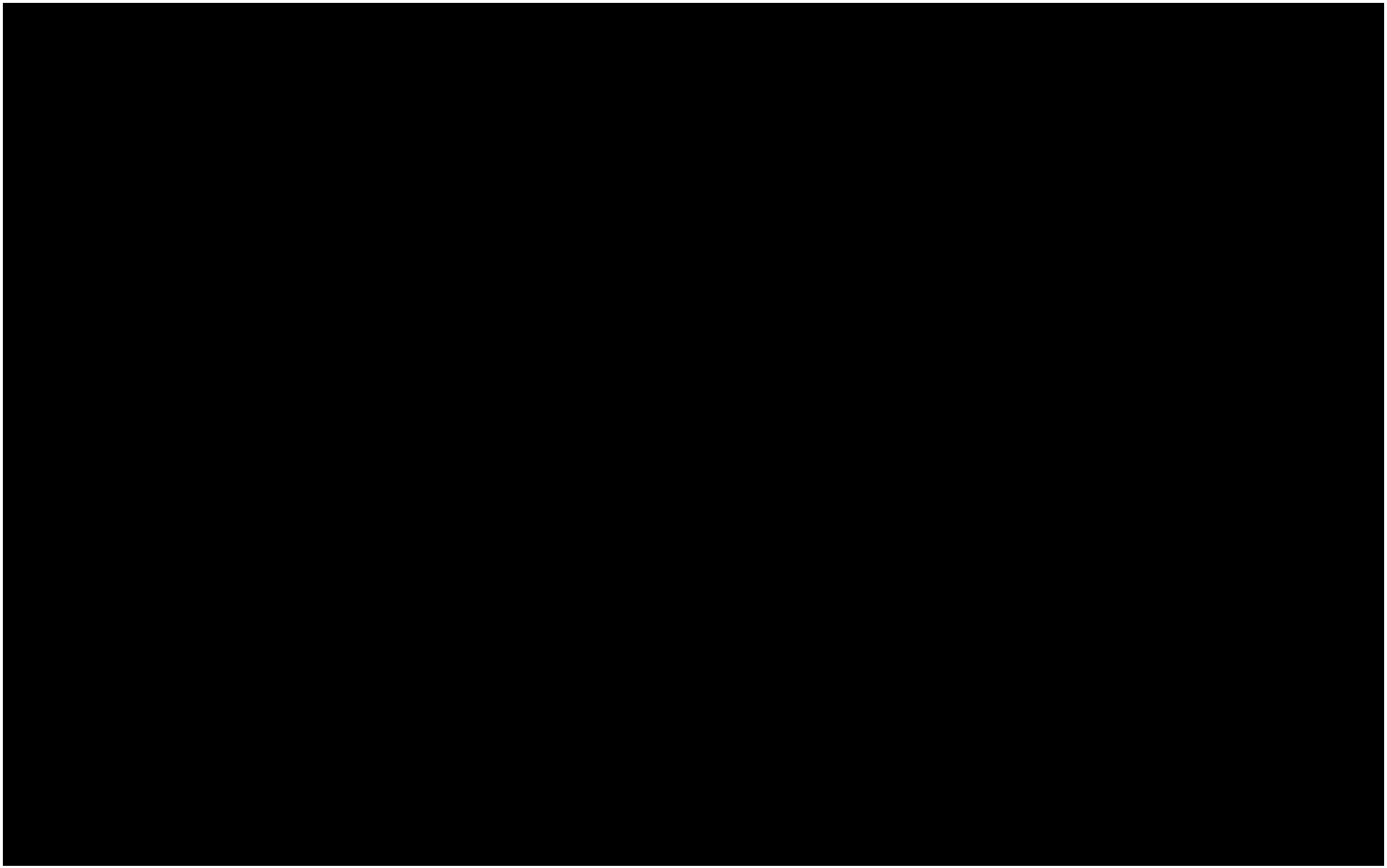
Data Processing Queue

```
-[NSOperationQueue cancelAllOperations]
```

Cancellation

Data Processing Queue

```
-[NSOperationQueue cancelAllOperations]
```



```
-[NSOperation isCancelled]
```

Cancellation

-[NSOperation isCancelled]

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[queue addOperationWithBlock:^{
    ...
}];
```

Cancellation

-[NSOperation isCancelled]

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
NSBlockOperation *op = [[NSBlockOperation alloc] init];
[op addExecutionBlock:^{
    ...
}];
[queue addOperation:op];
```

Cancellation

-[NSOperation isCancelled]

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
NSBlockOperation *op = [[NSBlockOperation alloc] init];
[op addExecutionBlock:^{
    for (int i = 0; i < 10000; i++) {
        processData(data[i]);
    }
}];
[queue addOperation:op];
```

Cancellation

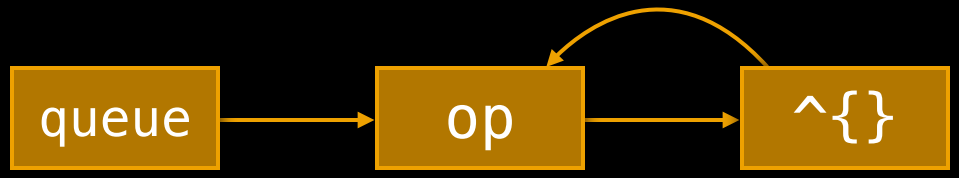
-[NSOperation isCancelled]

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
NSBlockOperation *op = [[NSBlockOperation alloc] init];
[op addExecutionBlock:^{
    for (int i = 0; i < 10000; i++) {
        if ([op isCancelled]) break;
        processData(data[i]);
    }
}];
[queue addOperation:op];
```

Cancellation

-[NSOperation isCancelled]

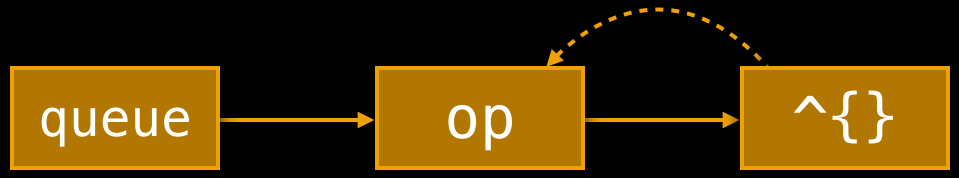
```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
NSBlockOperation *op = [[NSBlockOperation alloc] init];
[op addExecutionBlock:^{
    for (int i = 0; i < 10000; i++) {
        if ([op isCancelled]) break;
        processData(data[i]);
    }
}];
[queue addOperation:op];
```



Cancellation

-[NSOperation isCancelled]

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
NSBlockOperation *op = [[NSBlockOperation alloc] init];
__weak NSBlockOperation *weakOp = op;
[op addExecutionBlock:^{
    for (int i = 0; i < 10000; i++) {
        if ([weakOp isCancelled]) break;
        processData(data[i]);
    }
}];
[queue addOperation:op];
```



Demo

Cancellation

More Information

Jake Behrens
UI Frameworks Evangelist
behrens@apple.com

Documentation
Concurrency Programming Guide
http://developer.apple.com/library/ios

Apple Developer Forums
http://devforums.apple.com

Labs

iOS App Performance Lab	Developer Tools Lab A Wednesday 11:30AM
Cocoa Touch Lab	Essentials Lab B Thursday 9:00AM

What We've Learned

- Clear the queue with `-[NSOperationQueue cancelAllOperations]`
- Cancel a single operation with `-[NSOperationQueue cancel]`
- Those will not cancel running operations
- Check `-[NSOperation isCancelled]` in long-running operations
- Cancel table cell-related work in `tableView:didEndDisplayingCell:`


WWDC2012