Design Patterns

Singleton

- Объект, который создается только один раз и у которого есть глобальная точка доступа

```
@implementation Singleton

static Singleton *_sharedInstance = nil;

+ (Singleton *)sharedInstance {
    @synchronized(self) {
        if (!_sharedInstance) {
            _sharedInstance = [[Singleton alloc] init];
        }
    }
    return _sharedInstance;
}

@end
```

Singleton

- Правильный Singleton

```
@implementation Singleton

+ (Singleton *)sharedInstance {
    static dispatch_once_t pred;
    static Singleton *sharedInstance = nil;

    dispatch_once(&pred, ^{ sharedInstance = [[self alloc]
init]; });
    return sharedInstance;
}

@end
```

Delegate

- Объект (delegator object) вместо того, чтобы выполнять задачу самому, делегирует ее другому объекту (delegate object)

Delegate

- Определить protocol (@protocol DelegateProtocol <NSObject>)

- Добавить weak свойство (@property (weak) id<DelegateProtocol> delegate)

- Внутри делегатора вызвать соотвествующий метод делегата ([self.delegate performTask:task])

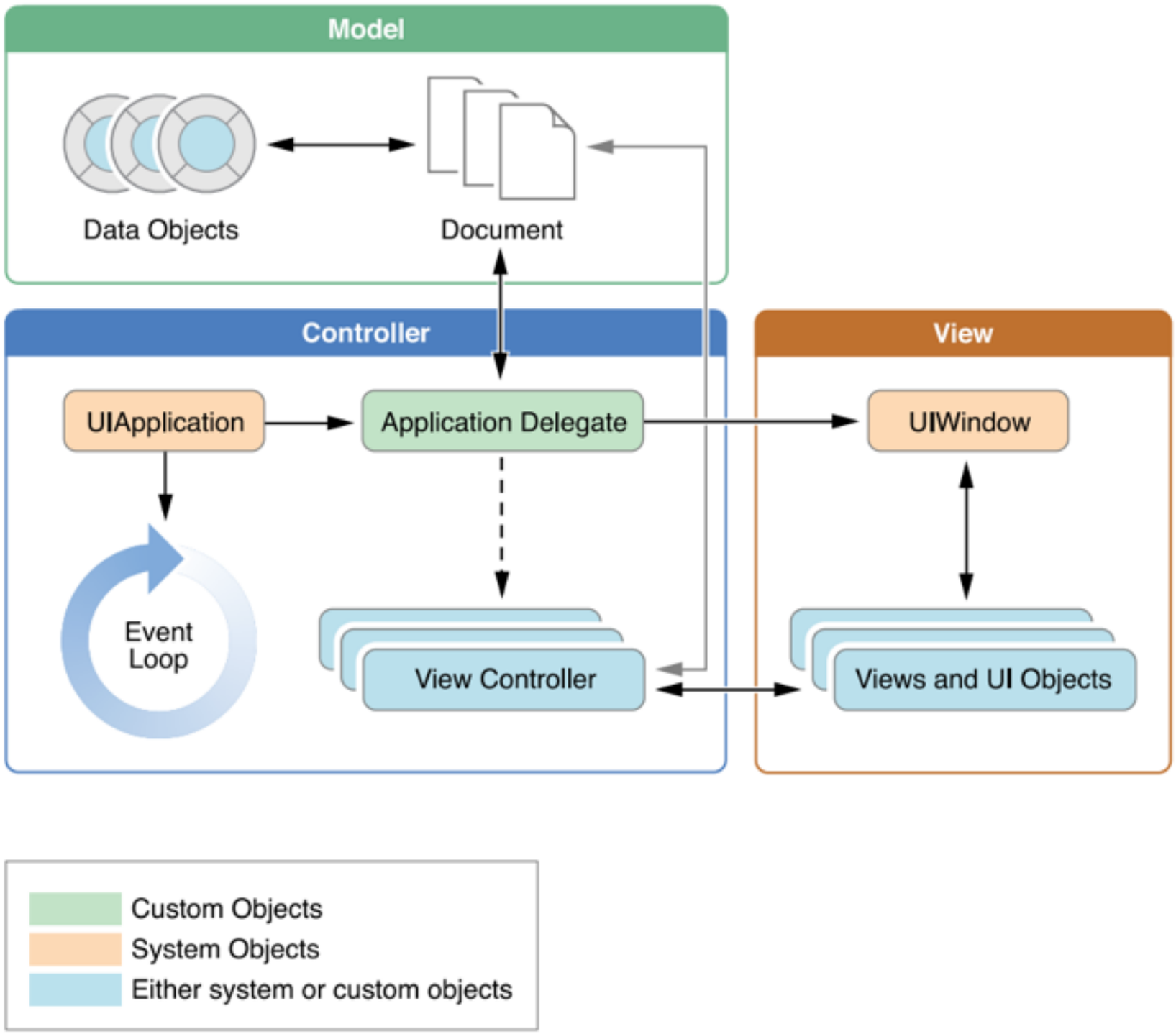
- Если метод делегата объявлен с @optional, то надо проверить существует метод или нет (if ([self.delegate respondToSelector:@(“performTask:”)

NSNotificationCenter

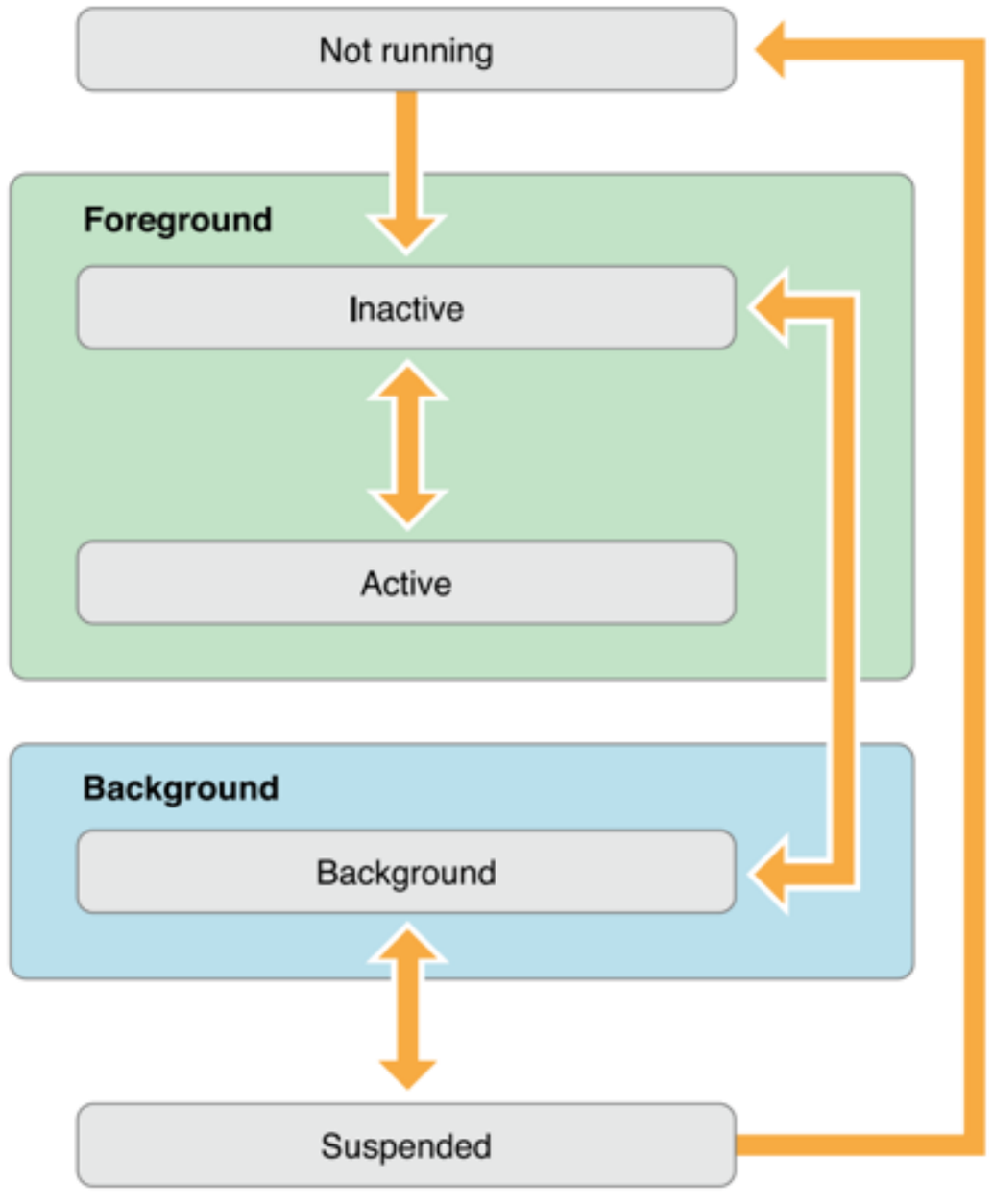
- Реализует паттерн наблюдатель
- Предоставляет механизм для обмена сообщениями внутри программы
- iOS шлет уведомления о изменениях ориентации приложения, появлении клавиатуры, смене состояния приложения, …

UIKit

App


Model
Data Objects
Document
Controller
UIApplication
Application Delegate
Event Loop
View Controller
View
UIWindow
Views and UI Objects
Custom Objects
System Objects
Either system or custom objects

States of App



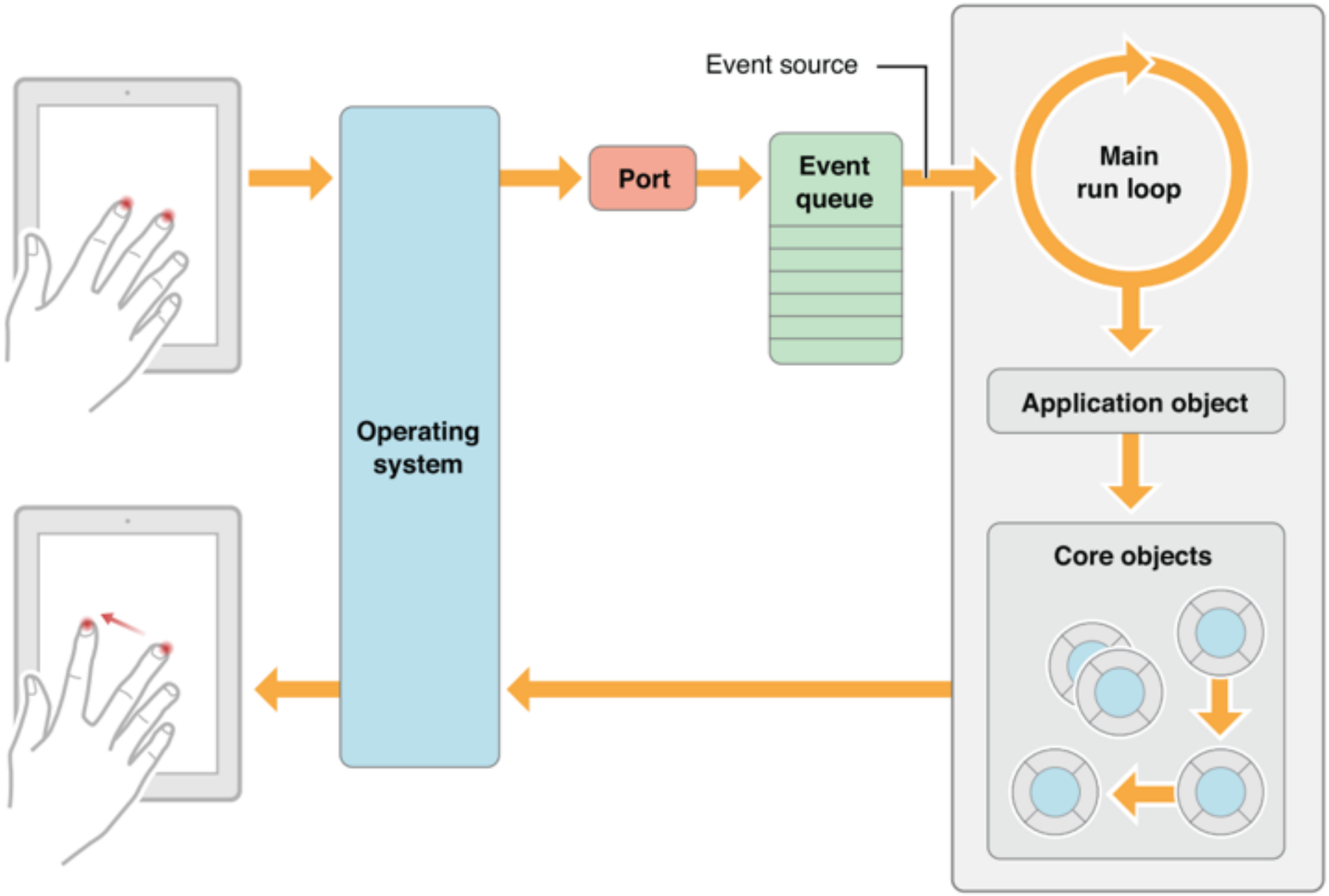
Run Loop

- Привязан к потоку
- Run Loop позволяет потоку выполнять несколько задач
- Представляет из себя цикл, в котором на каждой итерации проверяется очередь задач. Если задача есть, то поток начинает ее выполнение

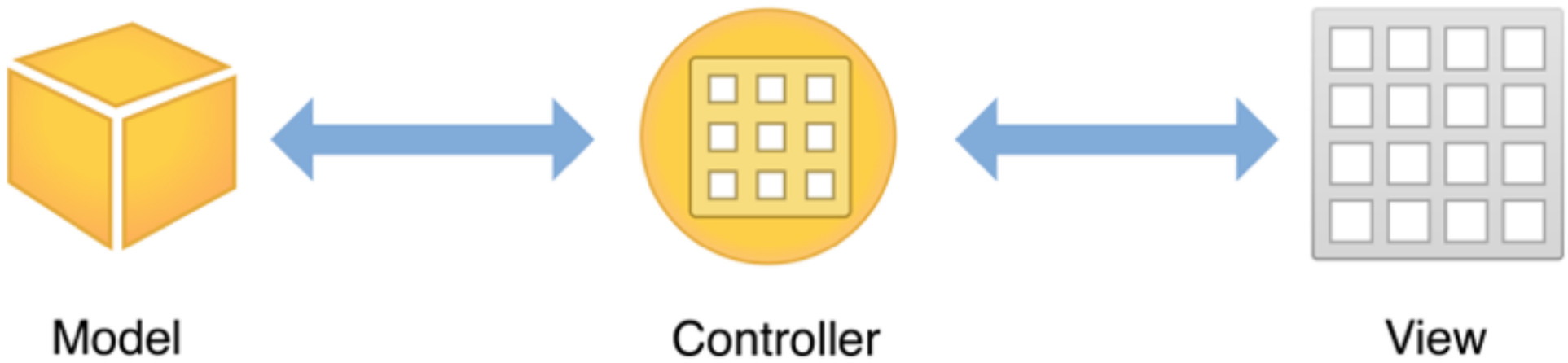
Main Run Loop

- Run Loop для главного потока (Main Thread)
- Обрабатывает все сообщения от пользователя

Main Run Loop



MVC

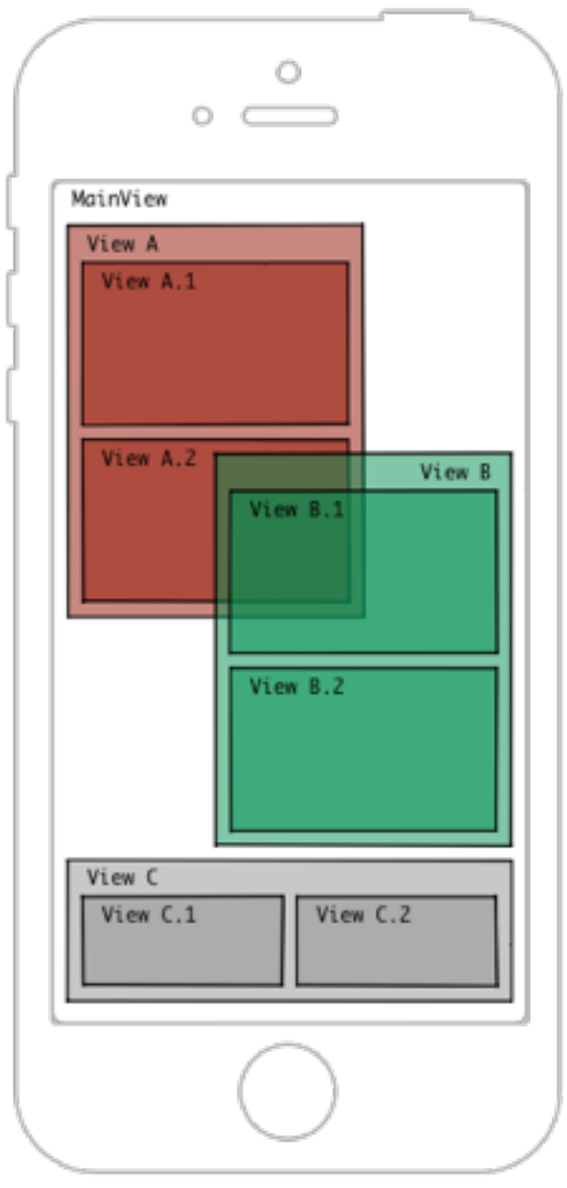


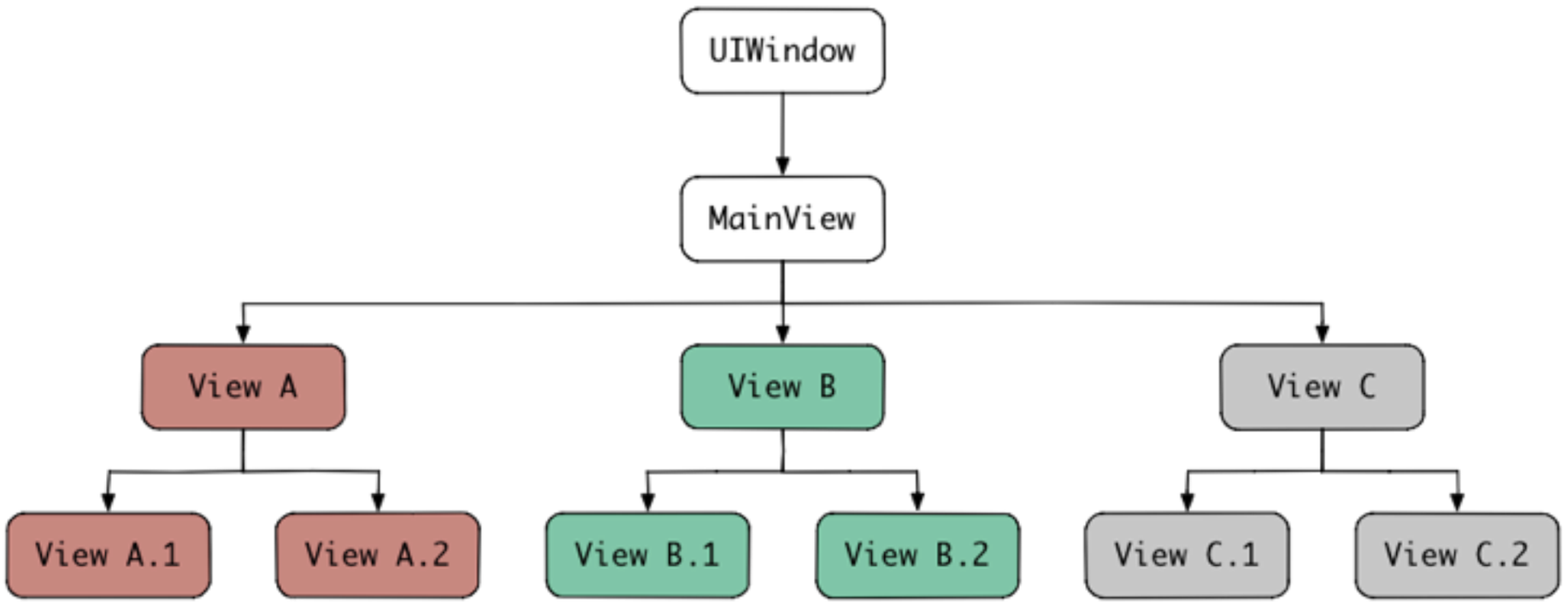
- Controller — наследники UIViewController
- View — наследники UIView
- Model — что угодно

UIView

- Определяет прямоугольную область на экране и определяет, что в ней будет отображаться
- Контейнер для других UIView
- Поле frame определяет позицию относительно родителя

UIView


MainView
View A
View A.1
View A.2
View B
View B.1
View B.2
View C
View C.1
View C.2


UIWindow
MainView
View A
View B
View C
View A.1
View A.2
View B.1
View B.2
View C.1
View C.2

UIViewController

- Управляет view
- Получает сообщения о появление/скрытие view
- Обрабатывает сообщения о повороте устройства
- Обрабатывает сообщение о нехватке памяти didReceiveMemoryWarning

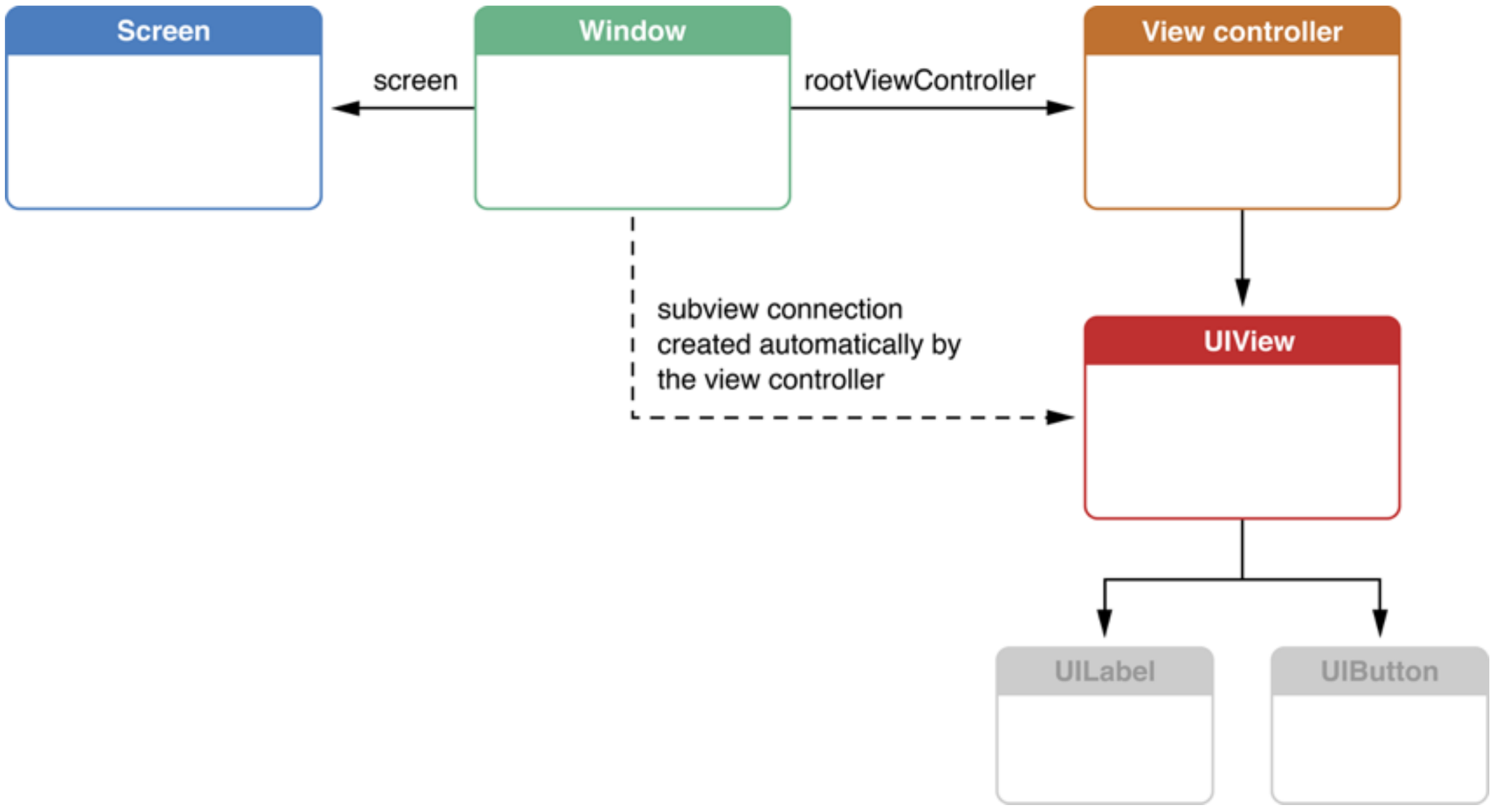
UIWindow

- Наследник UIView
- Содержит весь контент, отображаемые пользователю
- По умолчанию просто черный квадрат

UIWindow

- Может быть keyWindow и тогда будет получать сообщения от клавиатуры
- Свойство windowLevel определяет группу
- Свойство screen хранит UIScreen для этого окна
- Свойство rootViewController хранит UIViewController

UIWindow


Screen
Window
View controller
screen
rootViewController
subview connection
created automatically by
the view controller
UIView
UILabel
UIButton

UIScreen

- Содержит свойства ассоциированные с дисплеем (bounds, scale, …)
- Каждый UIWindow привязан к какому-нибудь окну

Main

```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate
                                                                     class]));
    }
}
```

UIApplicationMain

- Создает UIApplicaton
- Устанавливает делегат для UIApplicaton
- Устанавливает Run loop
- Загружает nib указанный в Info.plist

UIApplication

- Singleton
- Роутит сообщения от пользователя (touch, motion, control)
- Поддерживает иерархию окон (UIWindow)
- Управляет сообщениями
- ….

UIApplicationDelegate

- “Сердце” любого приложения
- Получает изменения о изменении состояния приложения
- Предоставляет UIWindow

UIEvent

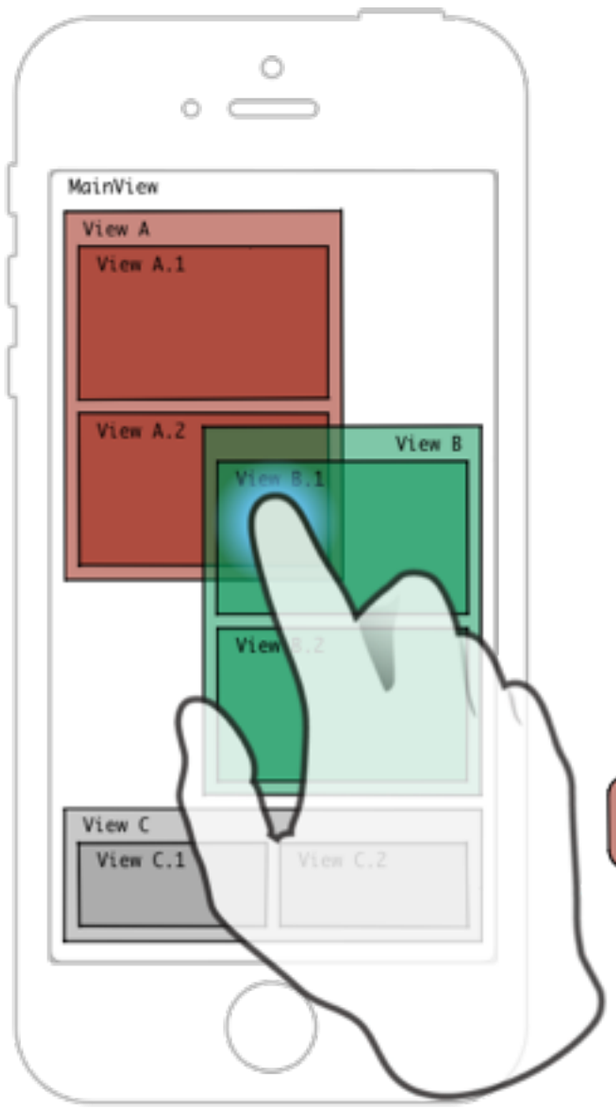
Может иметь тип

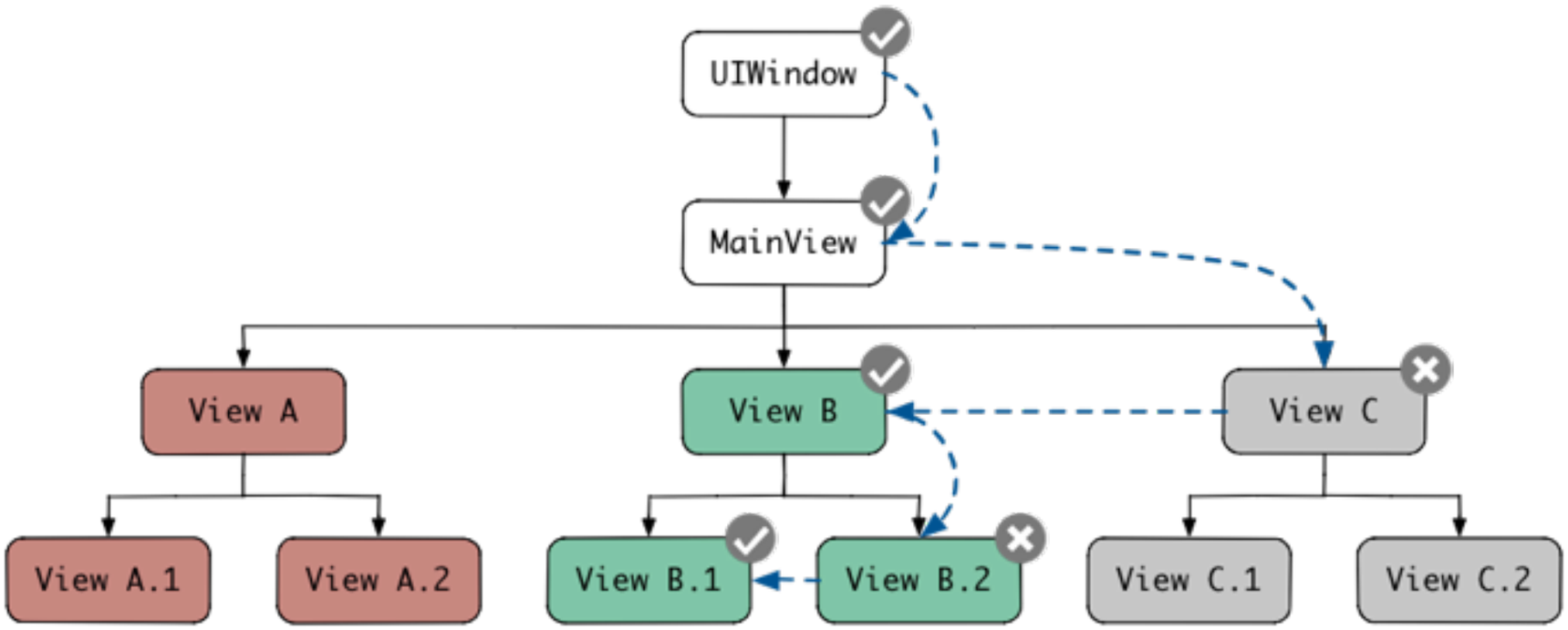
- touch
- motion
- remote-control

Event Delivery

- Пользовательский event попадает в Run Loop
- UIApplication пытается его обработать и отправляет key window
- Для touch evnets выполняется hit-test (поиск UIView в который попал touch)
- Для motion и remote control ищется first responder

Hit-Test


MainView
View A
View A.1
View A.2
View B
View B.1
View B.2
View C
View C.1
View C.2


UIWindow
MainView
View A
View B
View C
View A.1
View A.2
View B.1
View B.2
View C.1
View C.2

Hit-Test

- У каждой view вызывается hitTest:withEvent:

- Если hit-test view не может обрабтать event, то он идет по responder chain

Hit-Test

```
- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event {
    if (!self.isUserInteractionEnabled || self.isHidden || self.alpha <= 0.01) {
        return nil;
    }
    if ([self pointInside:point withEvent:event]) {
        for (UIView *subview in [self.subviews reverseObjectEnumerator]) {
            CGPoint convertedPoint = [subview convertPoint:point fromView:self];
            UIView *hitTestView = [subview hitTest:convertedPoint withEvent:event];
            if (hitTestView) {
                return hitTestView;
            }
        }
        return self;
    }
    return nil;
}
```

Responder Chain

- Последовательность связанных объектов, которые могут обработать какое-то сообщение
- Если объект не может обработать сообщение, то он пересылает его следующему объекту

UIResponder

- firstResponder получает первое сообщение

- Метод nextResponder возвращает следующий responder в цепочке

UIResponder

Что возвращает nextResponder

- UIView — UIViewController
- UIViewController — view.superview
- UIWindow — UIApplication
- UIApplication — nil

UIResponder

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;

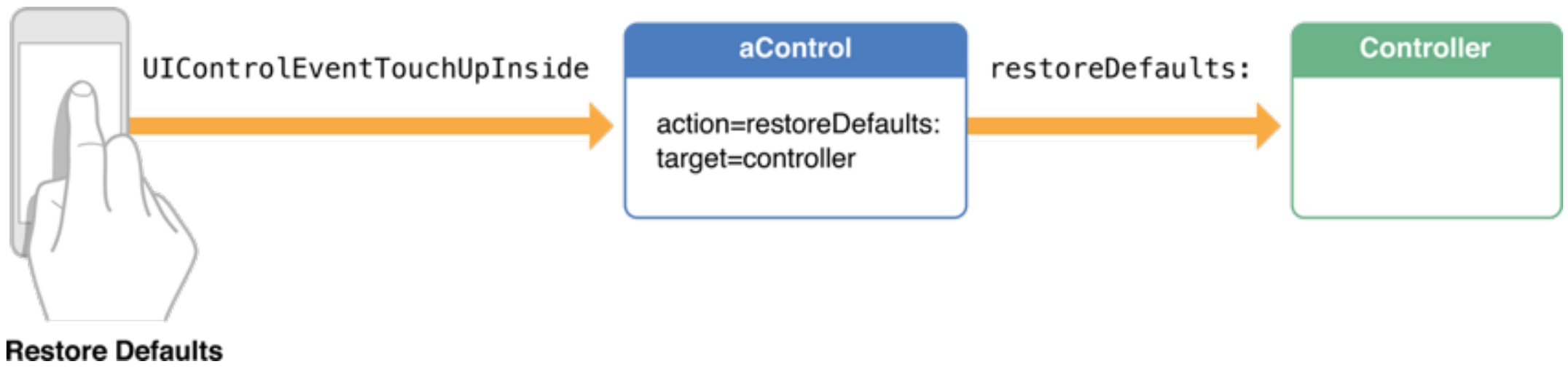
UIResponder

Реализация по умолчанию у UIView отправляет сообщение в nextResponder

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    // Forward this message up the responder chain
    [super touchesBegan:touches withEvent: event];
}
```

Target-Action

- Один объект шлет сообщение другому, когда происходит какое-то событие



UIControl

- Определяет интерфейс и базовую реализацию для Target-Action (addTarget, removeTarget, …)
- Позволяет управлять состоянием элемента (selected, enabled, highlighted, …)
- Сначала action идет в UIApplication
- Если action не реализован у target, то он идет по responder chain

UIControl

- UIButton
- UISegmentControl
- UISlider
- …

Gesture Recognizer

- Позволяют определить жест
- Добавляются к UIView
- Доступны Pan, Pinch, Tap, Swipe, Rotation, Long press

Threads

- Почти вся работа с UI должна выполняться на Main Thread
- При запуске (application:willFinishLaunchingWithOptions:) не стоит выполнять “тяжелые” задачи иначе приложение может быть убито системой

Background Taks

- Есть несколько секунд, чтобы выполнить какую-нибудь задачу при помощи beginBackgroundTaskWithName:expirationHandler: