

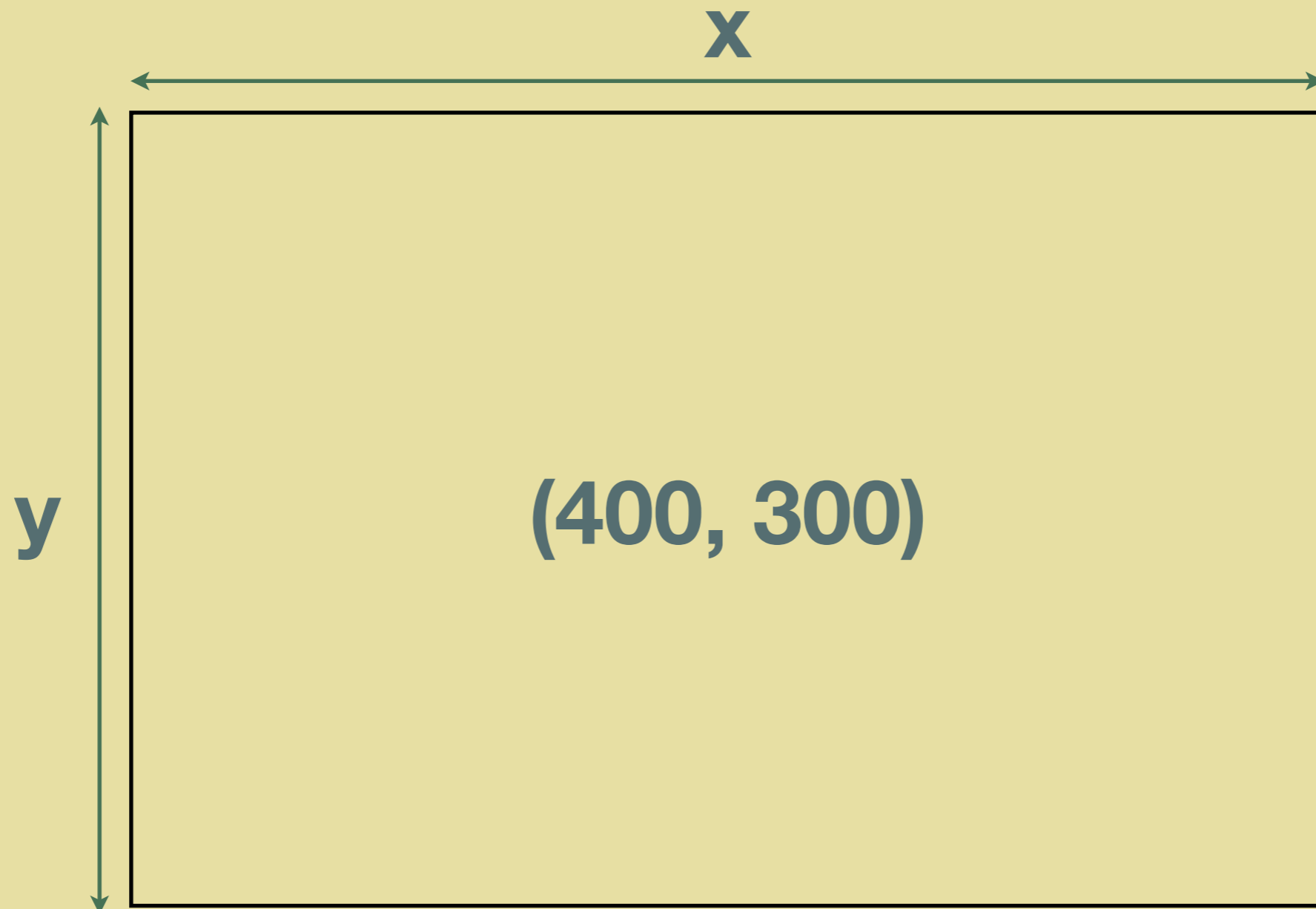
Pieter Omvlee - @pieteromvlee - Bohemian Coding

Drawing in UIKit

Drawing in UIKit

UIView, UIKit & CoreGraphics

View Basics



View Basics



View Basics

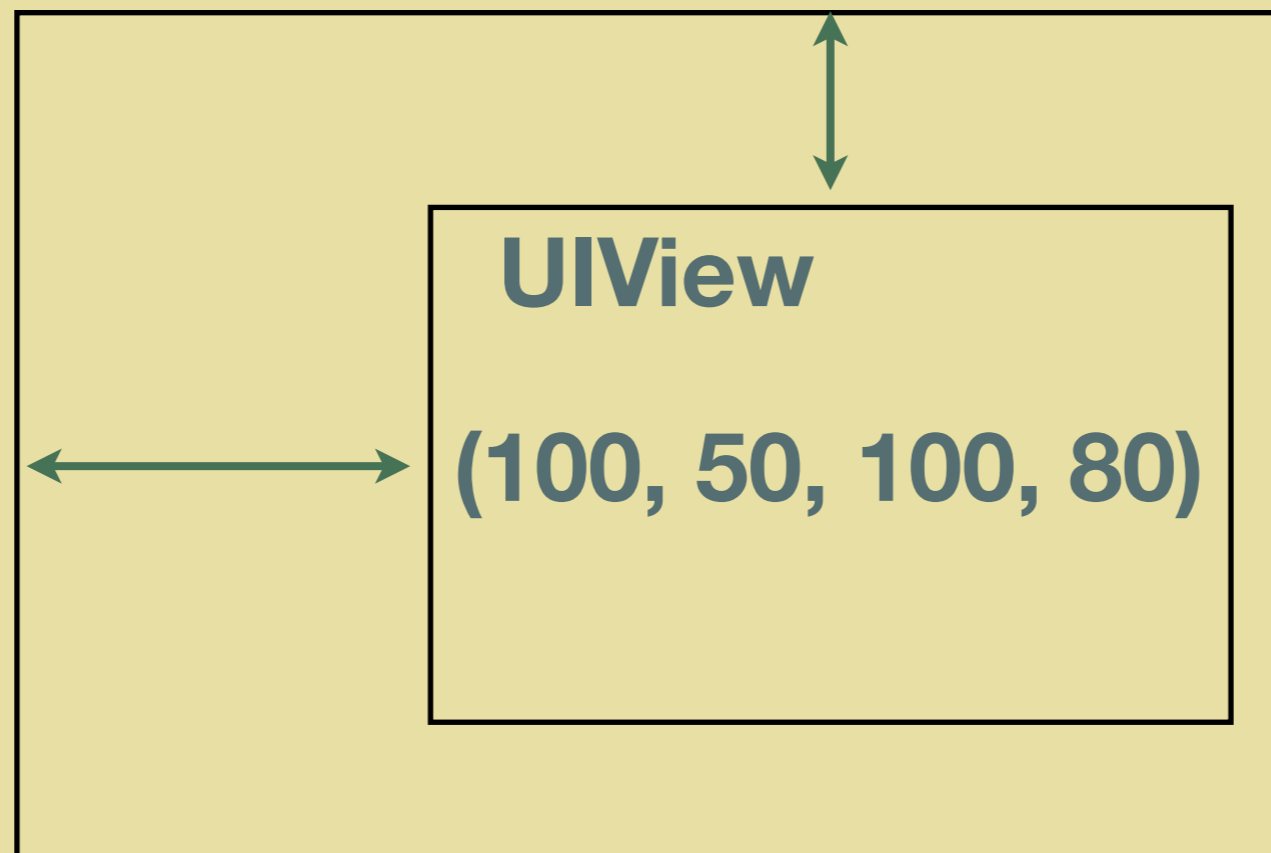
```
CGRect rect = CGRectMake(x, y, width, height);  
CGPoint origin = rect.origin;  
CGSize size = rect.size;
```

View Basics

```
[self bounds]  
[self frame];
```

View Basics

```
[self bounds]  
[self frame];
```



View Basics

```
[self bounds]  
[self frame];
```

UIView

(0, 0, 100, 80)

How?

You will get called

drawRect:

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    ...  
}
```

drawRect:

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    [[UIColor whiteColor] set];  
}
```

drawRect:

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    [[UIColor whiteColor] set];  
    UIRectFill([self bounds]);  
}
```

drawRect:

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    [[UIColor whiteColor] set];  
    UIRectFill([self bounds]);  
}
```

Set, then draw

Refresh

Refresh

```
[view setNeedsDisplay];
```

Refresh

```
[view setNeedsDisplay];
```

```
[view setNeedsDisplayInRect:CGRectMake(0,0,50,200)];
```

Refresh

```
[view setNeedsDisplay];
```

```
[view setNeedsDisplayInRect:CGRectMake(0,0,50,200)];
```



The diagram shows a large light green rectangle representing a container. Inside it is a smaller white rectangle representing a sub-view. The word "UIView" is written in bold dark blue text across the top of the white rectangle. A vertical white bar highlights the left edge of the white rectangle, indicating a specific region of interest.

UIView

Refresh

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    ...  
}
```

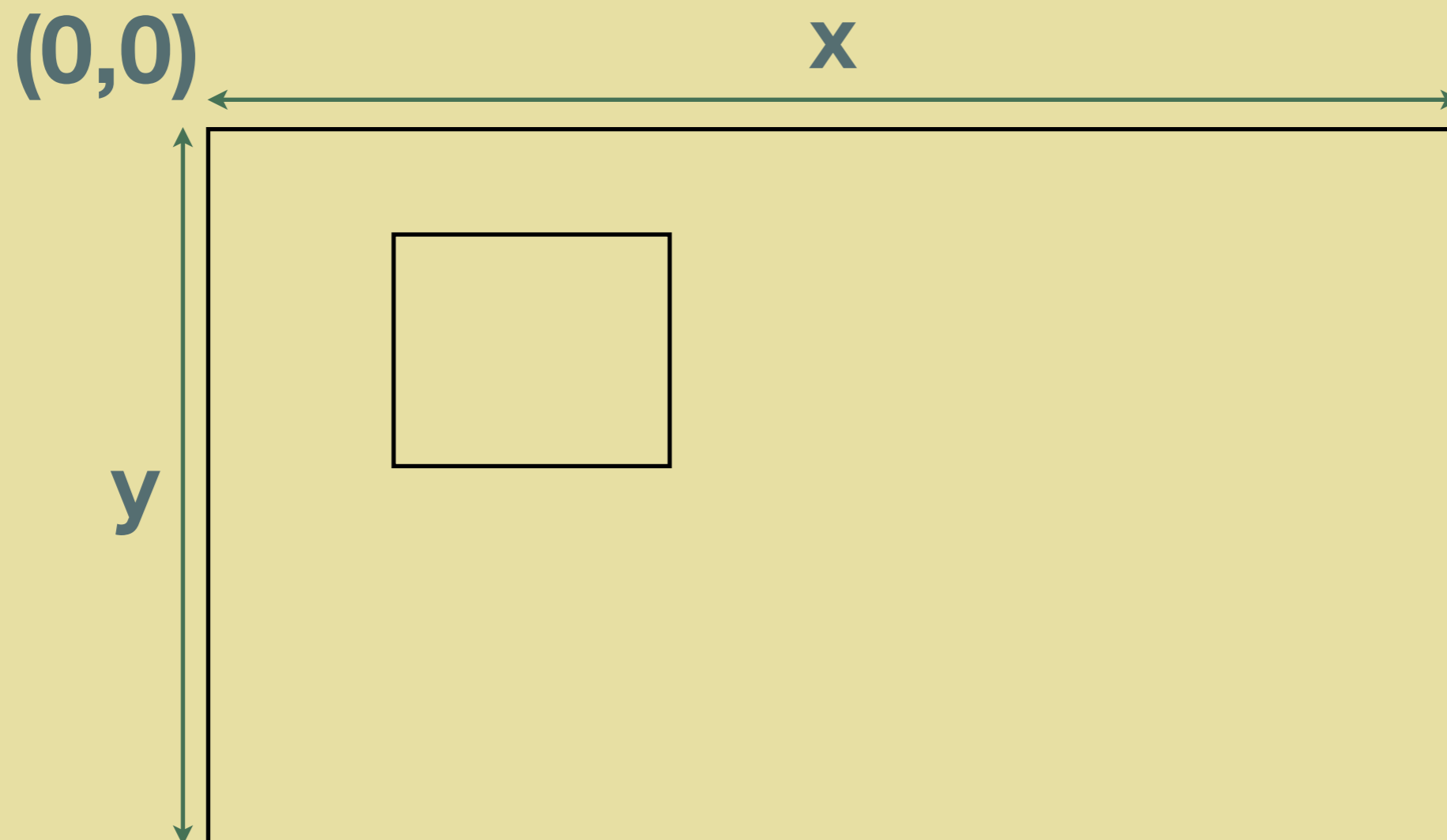
Refresh

```
[view setNeedsDisplayInRect:CGRectMake(50,50,80,80)];
```

```
[view setNeedsDisplayInRect:CGRectMake(150,150,80,80)];
```

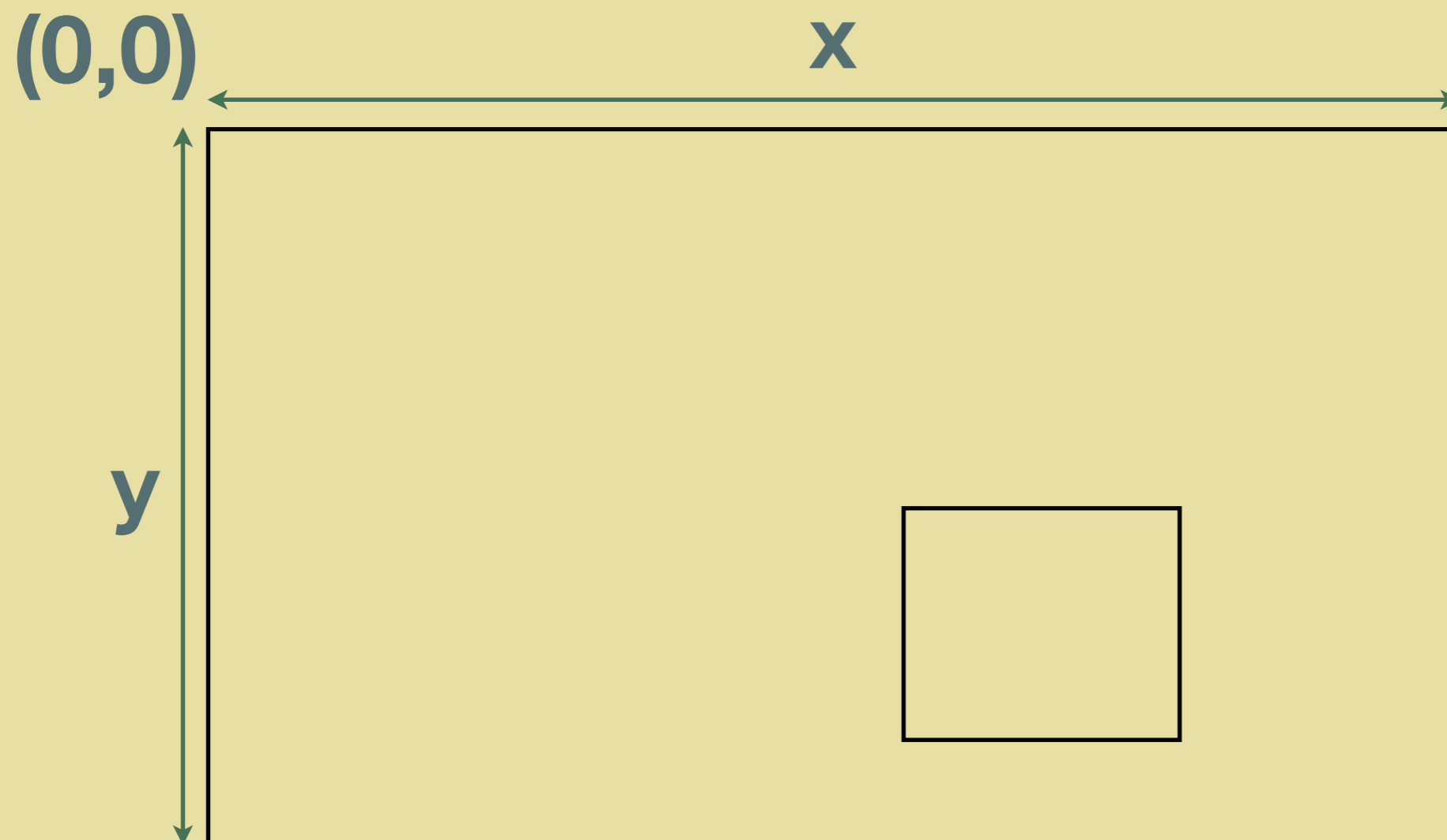
Refresh

```
[view setNeedsDisplayInRect:CGRectMake(50,50,80,80)];
```



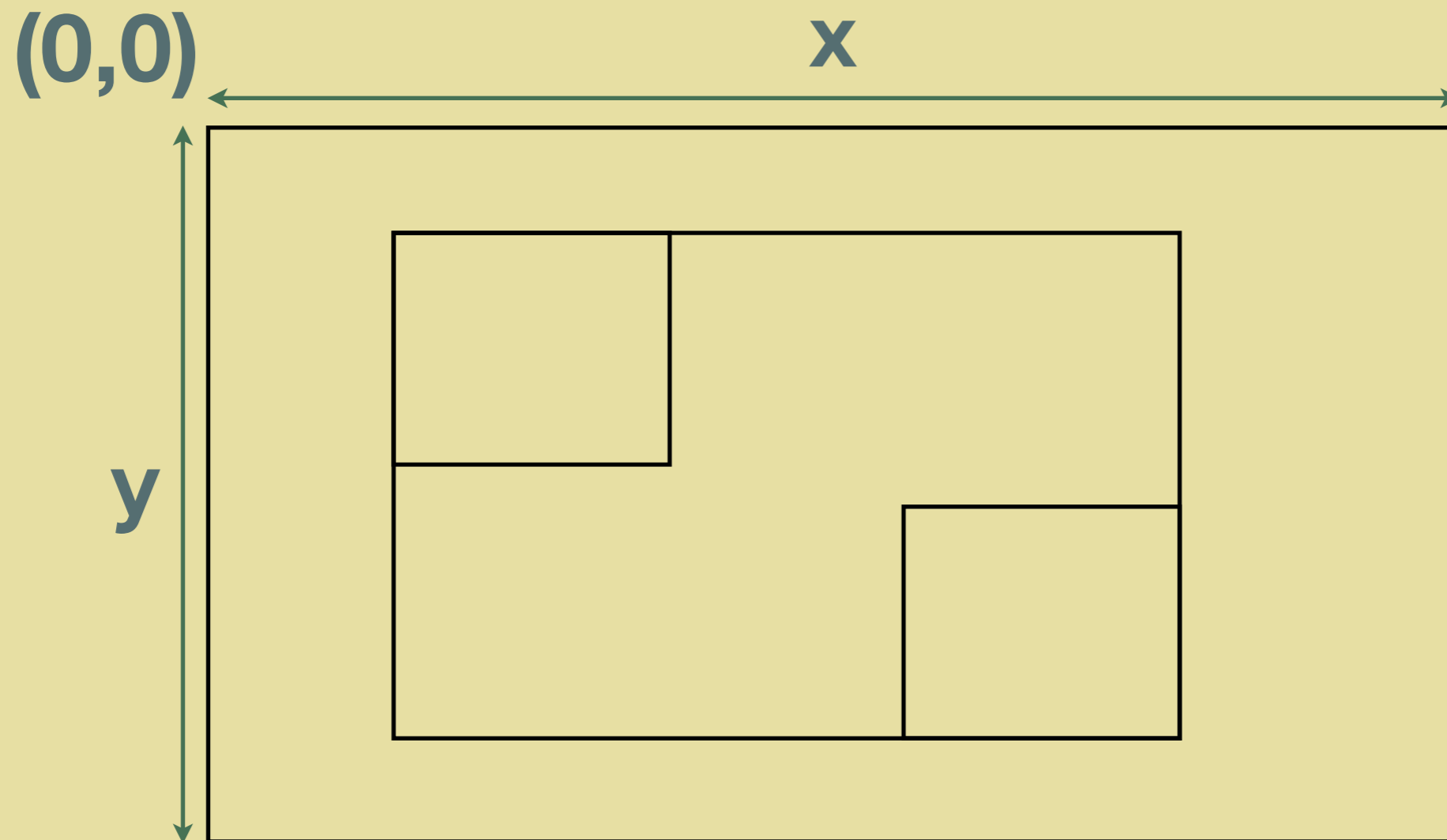
Refresh

```
[view setNeedsDisplayInRect:CGRectMake(150,150,80,80)];
```



Refresh

– (void)drawRect:(CGRect)dirtyRect



Expensive!
Refresh minimally!

-drawRect:

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    [[UIColor whiteColor] set];  
    UIRectFill([self bounds]);  
}
```

-drawRect:

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill([self bounds]);

    CGRect blackRect = CGRectMake(50, 50, 100, 100);
    [[UIColor blackColor] set];
    UIRectFill(blackRect);
}
```

-drawRect:

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill([self bounds]);

    CGRect blackRect = CGRectMake(50, 50, 100, 100);
    if (CGRectIntersectsRect(dirtyRect, blackRect)) {
        [[UIColor blackColor] set];
        UIRectFill(blackRect);
    }
}
```

-drawRect:

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill(dirtyRect);

    CGRect blackRect = CGRectMake(50, 50, 100, 100);
    if (CGRectIntersectsRect(dirtyRect, blackRect)) {
        [[UIColor blackColor] set];
        UIRectFill(blackRect);
    }
}
```

Expensive!

Refresh minimally, **Draw** minimally

-drawRect:

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill(dirtyRect);

    CGRect blackRect = CGRectMake(50, 50, 100, 100);
    if (CGRectIntersectsRect(dirtyRect, blackRect)) {
        [[UIColor blackColor] set];
        UIRectFill(blackRect);
    }
}
```

[self bounds] != dirtyRect

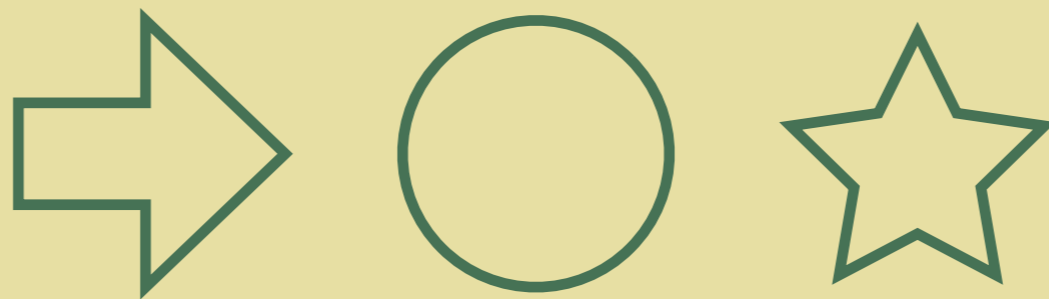
And now for something completely different....

UIBezierPath

CGPath wrapper

UIBezierPath

CGPath wrapper

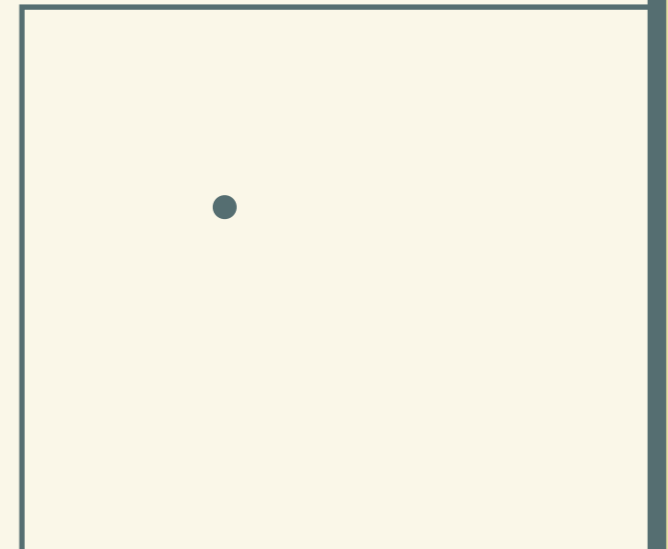


UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    UIBezierPath *bezier = [UIBezierPath bezierPath];  
}
```

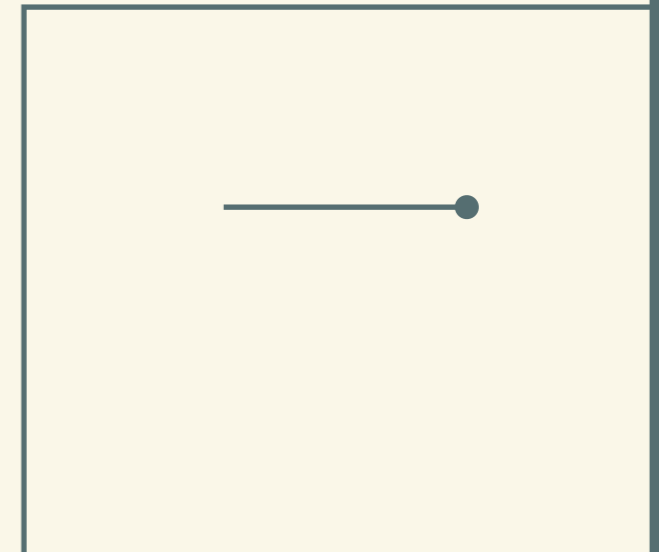
UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    UIBezierPath *bezier = [UIBezierPath bezierPath];  
    [bezier moveToPoint:CGPointMake(100,100)];  
}
```



UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    UIBezierPath *bezier = [UIBezierPath bezierPath];  
    [bezier moveToPoint:CGPointMake(100,100)];  
    [bezier addLineToPoint:CGPointMake(200,100)];  
}
```



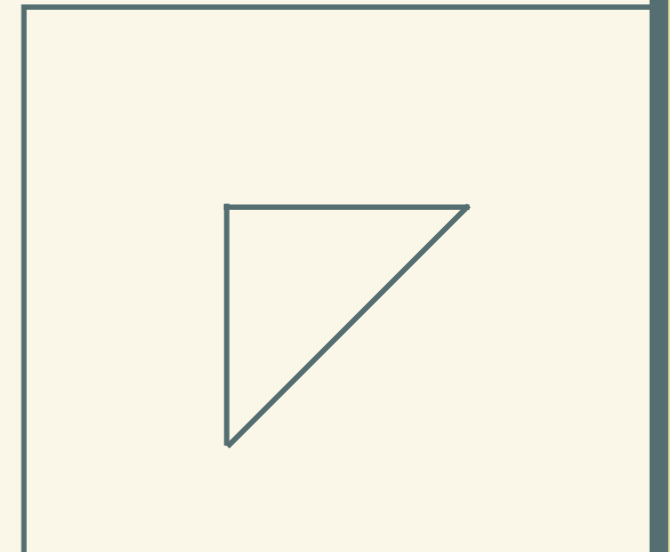
UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIBezierPath *bezier = [UIBezierPath bezierPath];
    [bezier moveToPoint:CGPointMake(100,100)];
    [bezier addLineToPoint:CGPointMake(200,100)];
    [bezier addLineToPoint:CGPointMake(100,200)];
}
```



UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIBezierPath *bezier = [UIBezierPath bezierPath];
    [bezier moveToPoint:CGPointMake(100,100)];
    [bezier addLineToPoint:CGPointMake(200,100)];
    [bezier addLineToPoint:CGPointMake(100,200)];
    [bezier closePath];
}
```



UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIBezierPath *bezier = [UIBezierPath bezierPath];
    [bezier moveToPoint:CGPointMake(100,100)];
    [bezier addLineToPoint:CGPointMake(200,100)];
    [bezier addLineToPoint:CGPointMake(100,200)];
    [bezier closePath];

    [[UIColor redColor] set];
}
```

UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIBezierPath *bezier = [UIBezierPath bezierPath];
    [bezier moveToPoint:CGPointMake(100,100)];
    [bezier addLineToPoint:CGPointMake(200,100)];
    [bezier addLineToPoint:CGPointMake(100,200)];
    [bezier closePath];

    [[UIColor redColor] set];
    [bezier fill];
}
```

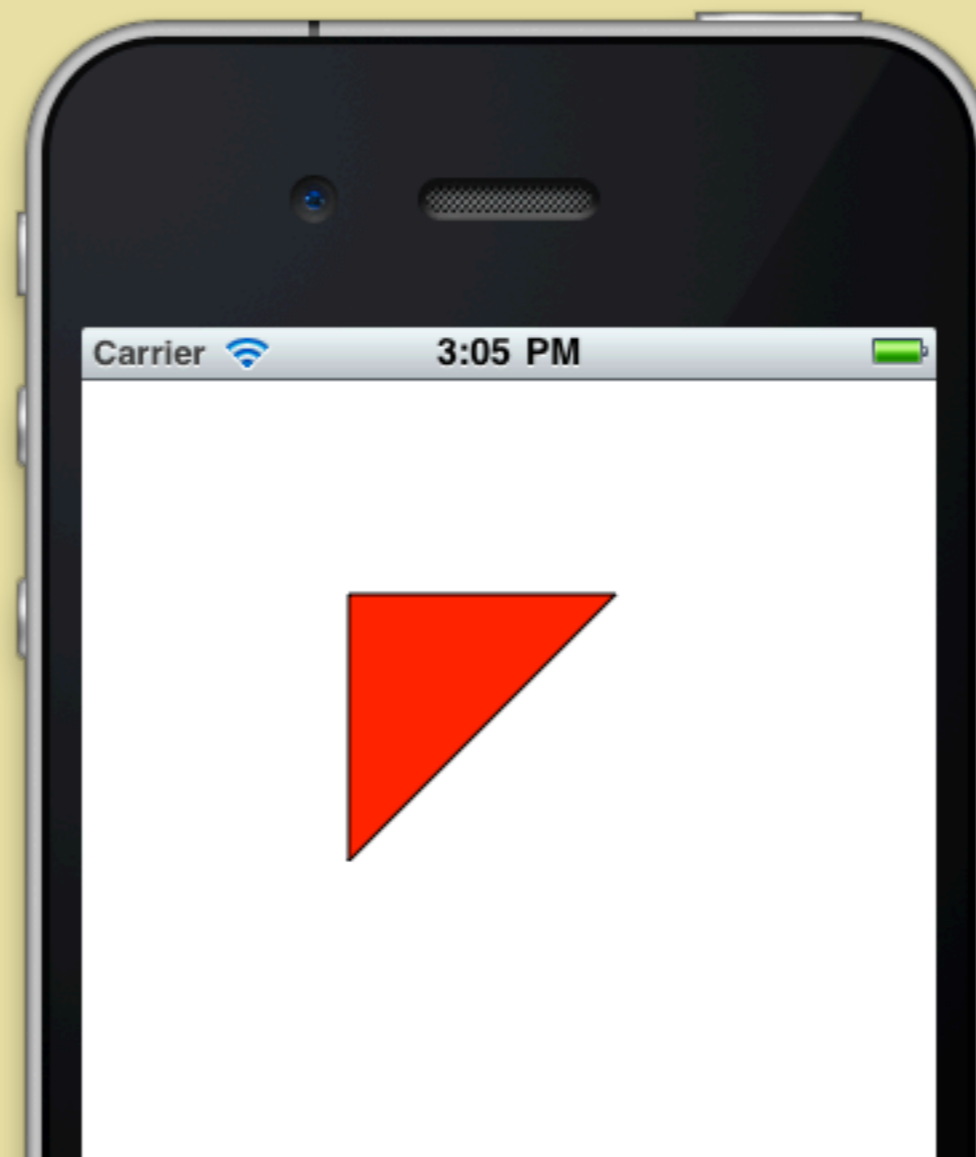
UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIBezierPath *bezier = [UIBezierPath bezierPath];
    [bezier moveToPoint:CGPointMake(100,100)];
    [bezier addLineToPoint:CGPointMake(200,100)];
    [bezier addLineToPoint:CGPointMake(100,200)];
    [bezier closePath];

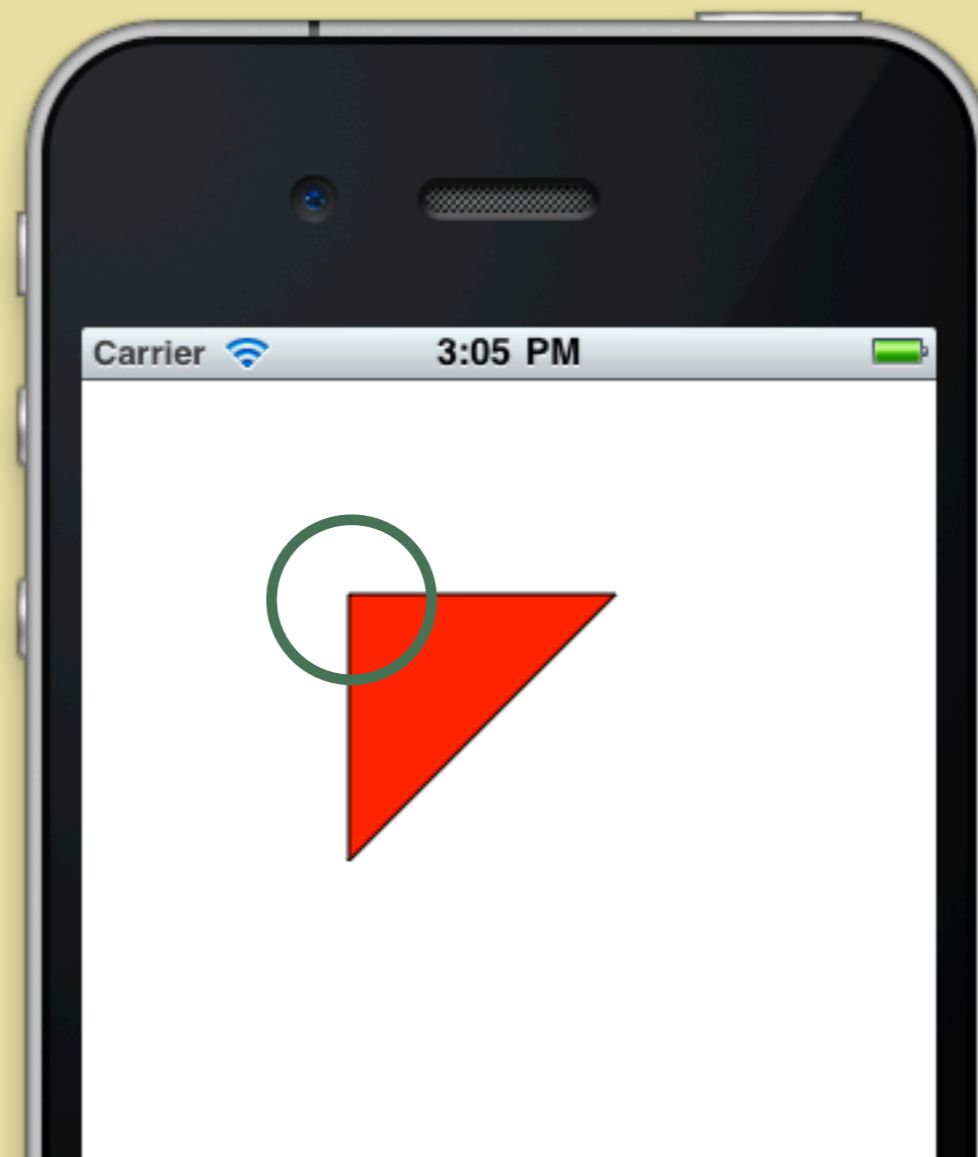
    [[UIColor redColor] set];
    [bezier fill];

    [[UIColor blackColor] set];
    [bezier stroke];
}
```

UIBezierPath



Fuzzy Paths



Fuzzy Paths

Fuzzy Paths

		(2,2)					

Fuzzy Paths

		(2,2)				(6,2)	

Fuzzy Paths

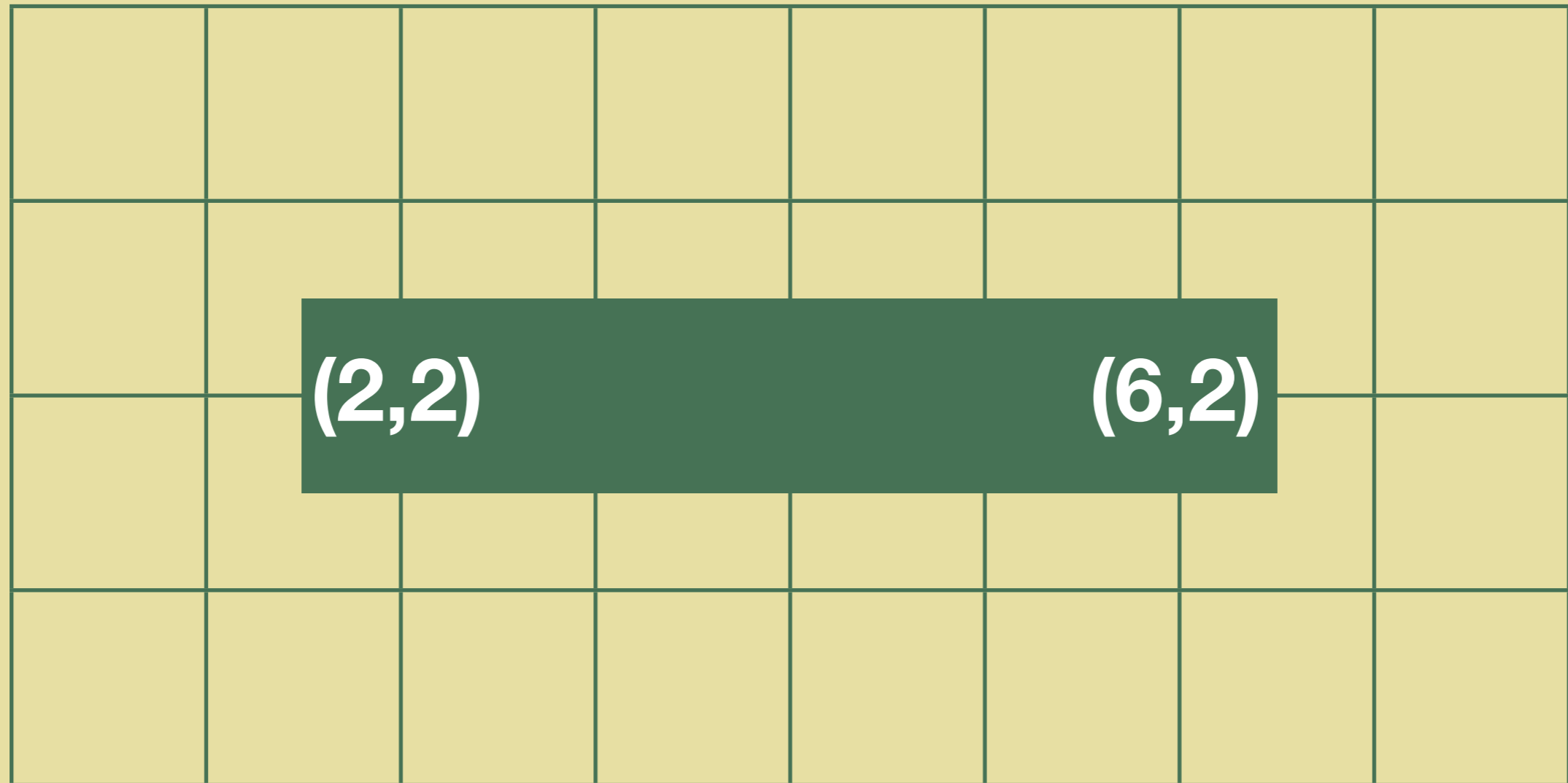
		(2,2)				(6,2)	

Fuzz Paths

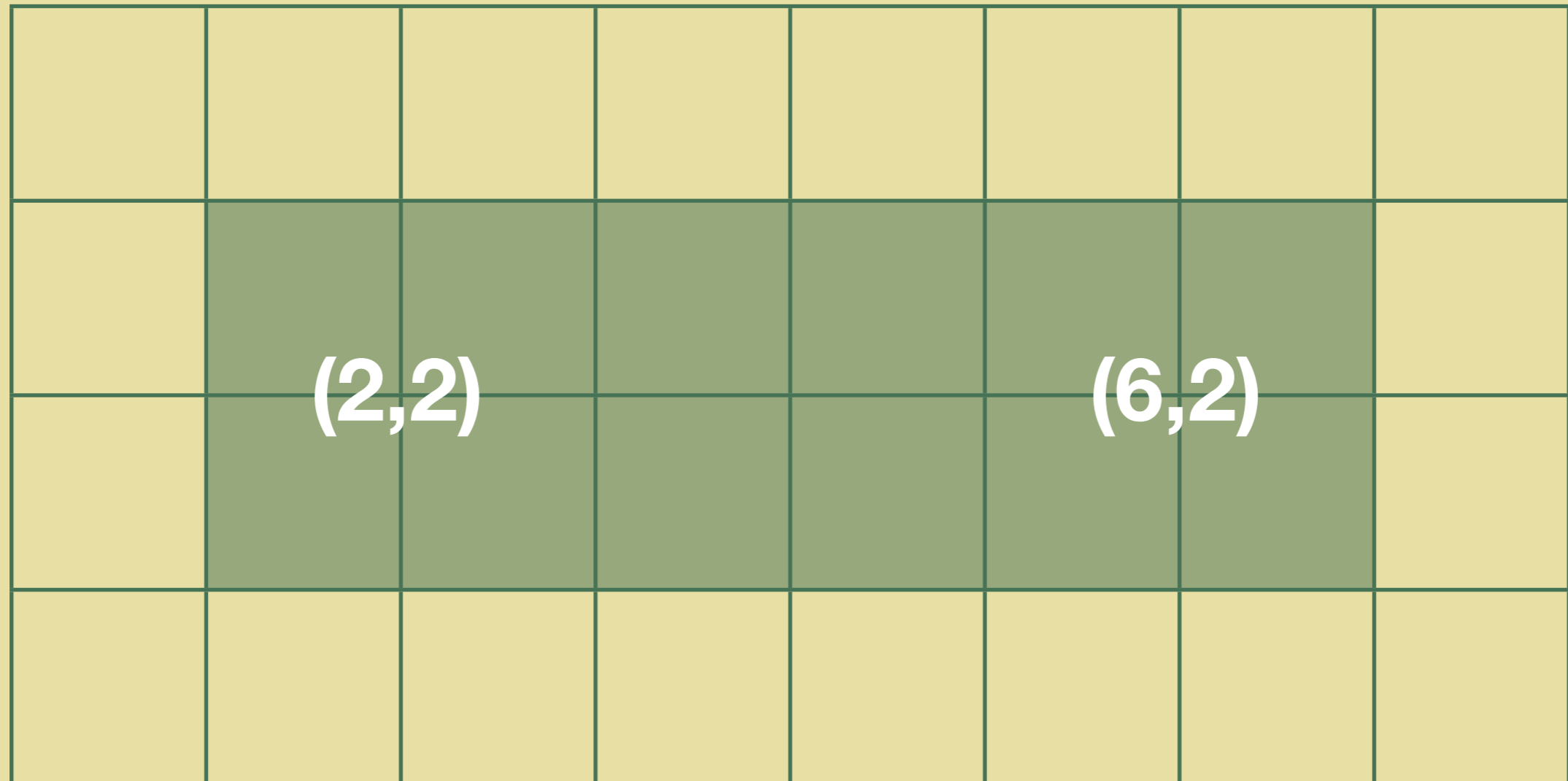


Fuzzy Paths

Fuzzy Paths



Fuzzy Paths



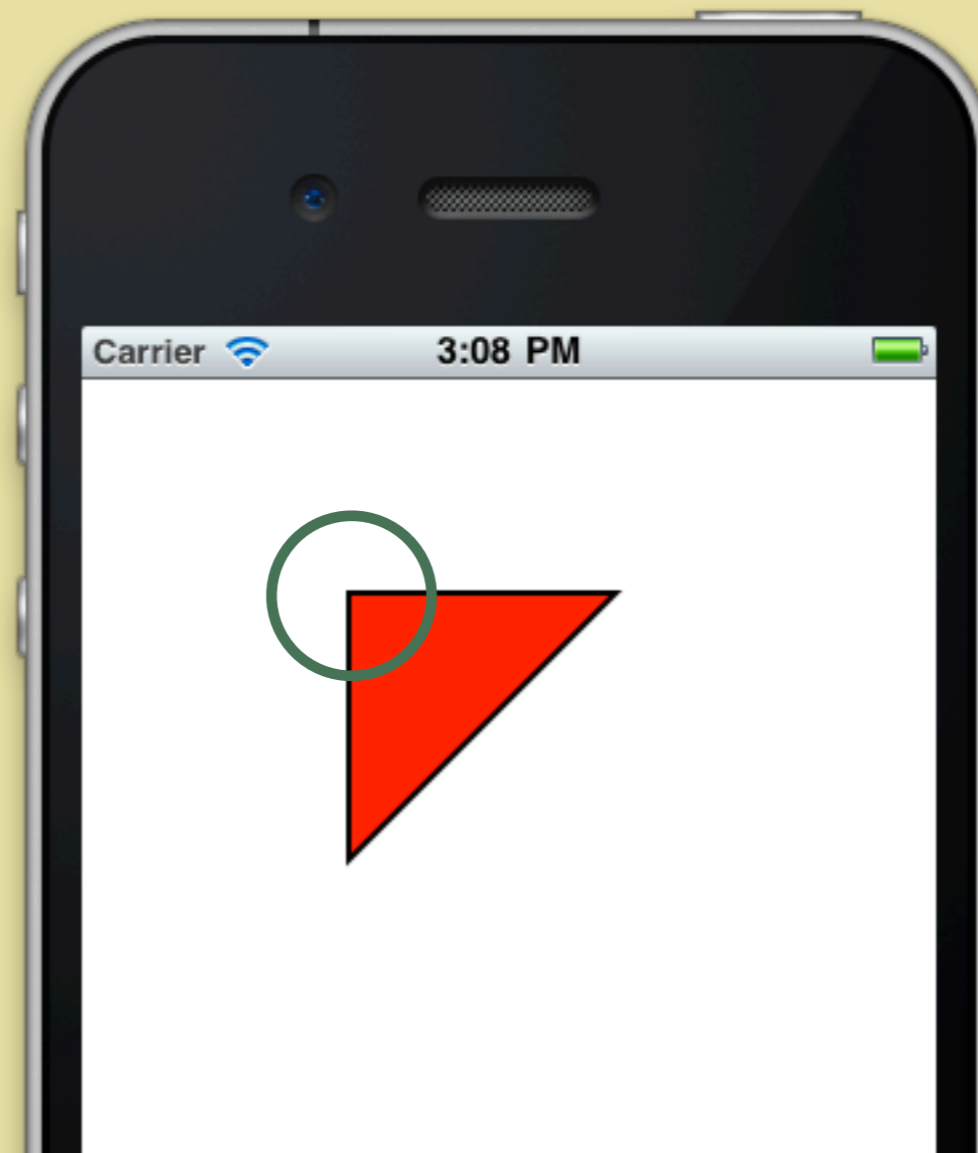
UIBezierPath

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIBezierPath *bezier = [UIBezierPath bezierPath];
    [bezier moveToPoint:CGPointMake(100,100)];
    [bezier addLineToPoint:CGPointMake(200,100)];
    [bezier addLineToPoint:CGPointMake(100,200)];
    [bezier closePath];

    [[UIColor redColor] set];
    [bezier fill];

    bezier.lineWidth = 2;
    [[UIColor blackColor] set];
    [bezier stroke];
}
```

Fuzzy Paths



offset by 0.5px

CGAffineTransform

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill(dirtyRect);

    [[UIColor redColor] set];

    CGContextTranslateCTM(
        UIGraphicsGetCurrentContext(),
        0.5, 0.5);

    [[UIBezierPath bezierPathWithRect:
        CGRectMake(50, 50, 100, 100)] stroke];
}
```

UI Image

UIImage

```
- (void)drawRect:(CGRect)dirtyRect
{
    UIImage *image = [UIImage imageNamed:@"..."];
    [image drawAtPoint:CGPointMake(50, 50)];
}
```

NSString

NSString

```
- (void)drawRect:(CGRect)dirtyRect
{
    NSString *string = @"This is a string";
    [[UIColor redColor] set];
    [string drawAtPoint:CGPointMake(50, 50) withFont:
        [UIFont systemFontOfSize:18]];
}
```

GraphicsContext

GraphicsContext

Acts like a **Stack**

GraphicsContext

```
CGContextSaveGState(...)
```

```
...
```

```
CGContextRestoreGState(...)
```

Expensive!

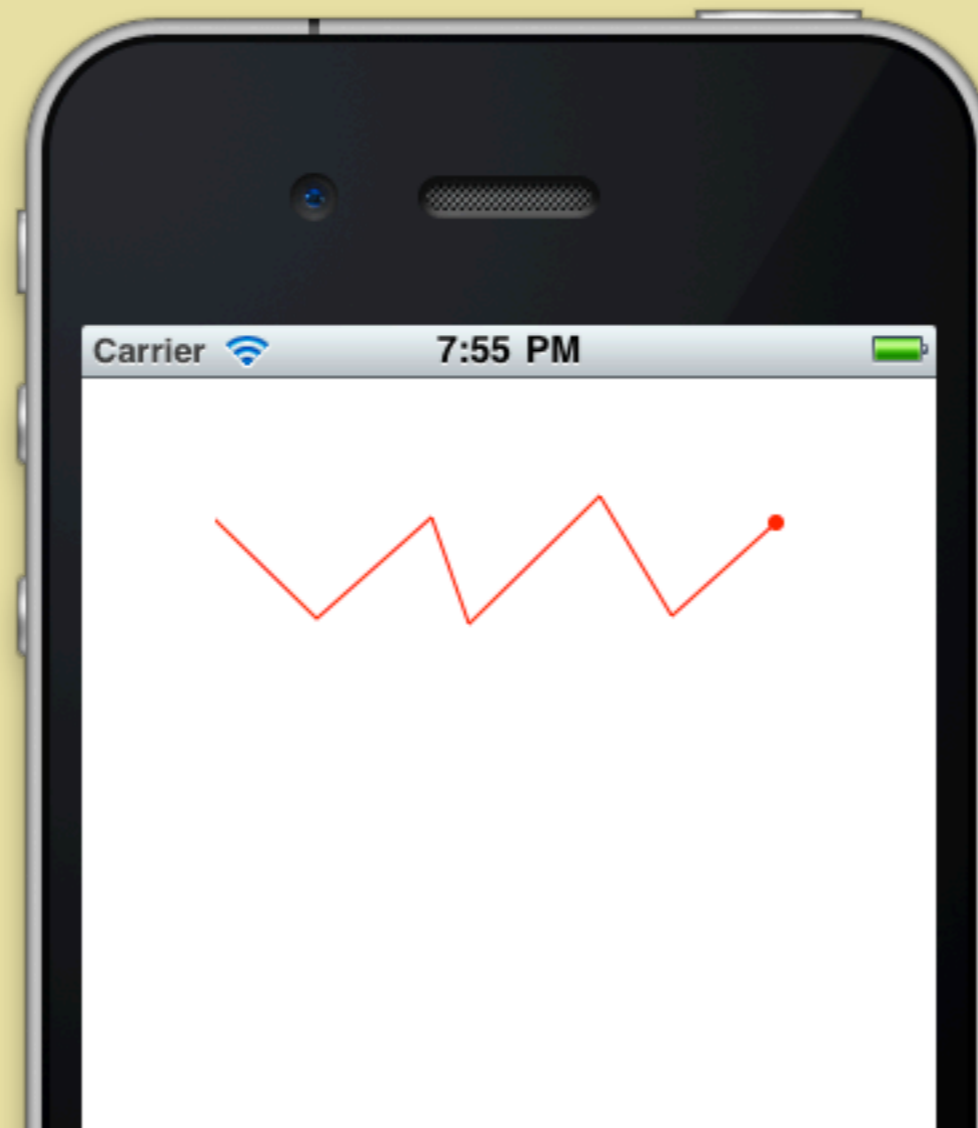
Refresh minimally, **Draw** minimally

Expensive!

Refresh minimally, Draw minimally & don't create superfluous CGContexts

UIEvent

UIEvent



UIEvent

```
- (void)touchesBegan:(NSSet *)touches  
    withEvent:(UIEvent *)event  
{  
  
}
```

UIEvent

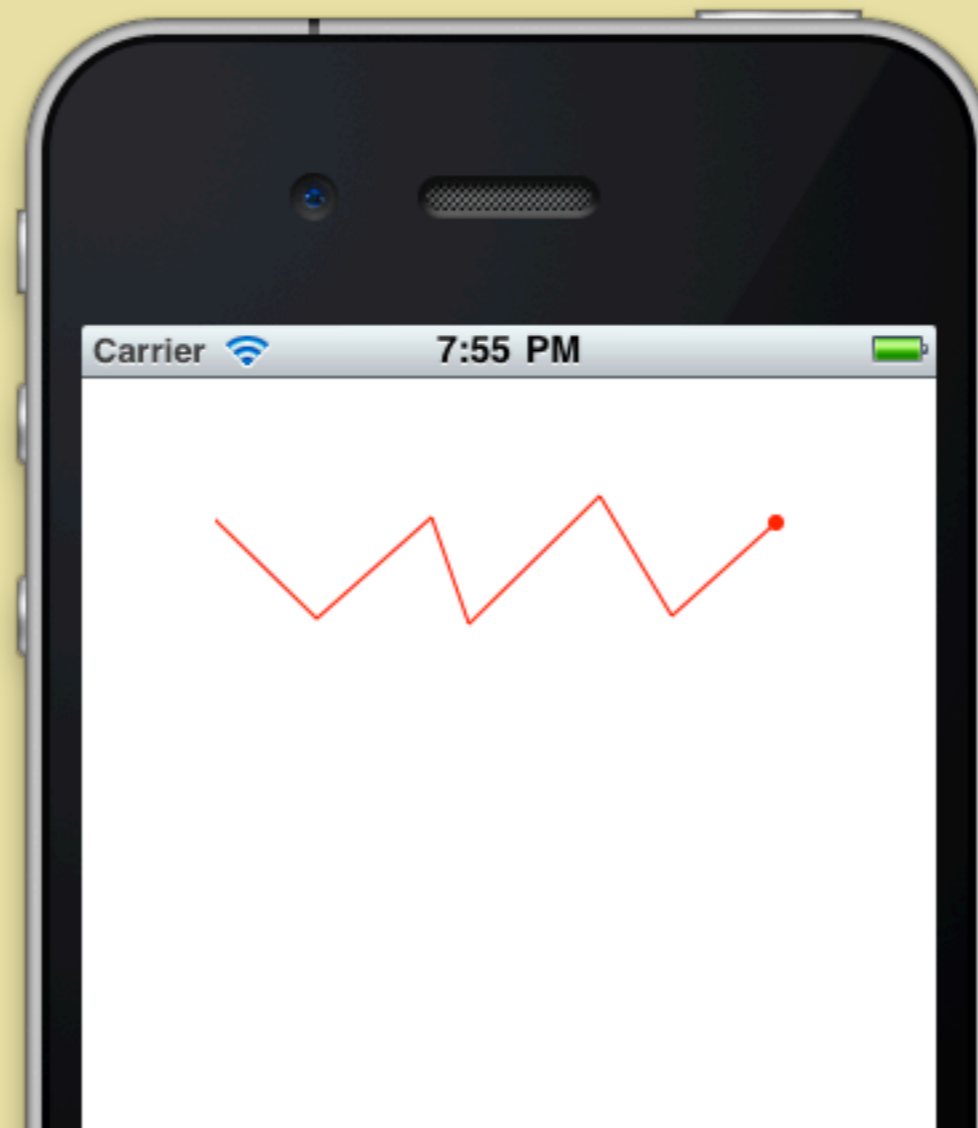
```
- (void)touchesBegan:(NSSet *)touches  
    withEvent:(UIEvent *)event  
{  
    CGPoint point = [[touches anyObject] locationInView:self];  
}
```

UIEvent

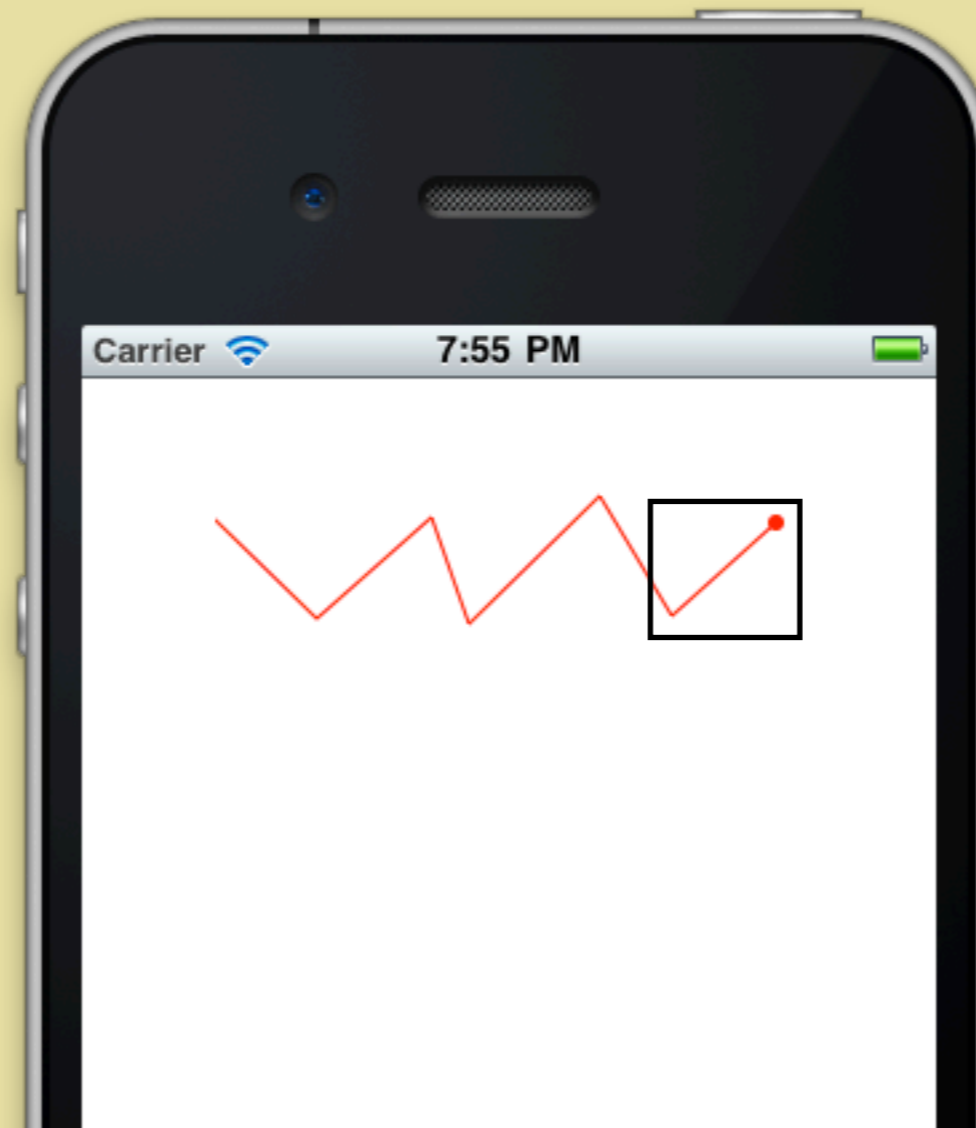
```
- (void)touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event
{
    CGPoint point = [[touches anyObject] locationInView:self];

    if ([bezierPath isEmpty])
        [bezierPath moveToPoint:point];
    else
        [bezierPath addLineToPoint:point];
}
```

UIEvent



UIEvent



UIEvent

```
- (void)touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event
{
    CGPoint point = [[touches anyObject] locationInView:self];
    CGPoint oldPoint = [bezierPath currentPoint];

    if ([bezierPath isEmpty])
        [bezierPath moveToPoint:point];
    else
        [bezierPath addLineToPoint:point];
}
```

UIEvent

```
- (void)touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event
{
    CGPoint point = [[touches anyObject] locationInView:self];
    CGPoint oldPoint = [bezierPath currentPoint];

    if ([bezierPath isEmpty])
        [bezierPath moveToPoint:point];
    else
        [bezierPath addLineToPoint:point];

    CGRect dirtyRect = [self rectBetweenPoint:point
                                   andPoint:oldPoint];
}
```

UIEvent

```
- (void)touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event
{
    CGPoint point = [[touches anyObject] locationInView:self];
    CGPoint oldPoint = [bezierPath currentPoint];

    if ([bezierPath isEmpty])
        [bezierPath moveToPoint:point];
    else
        [bezierPath addLineToPoint:point];

    CGRect dirtyRect = [self rectBetweenPoint:point
                                         andPoint:oldPoint];
    [self setNeedsDisplayInRect:dirtyRect];
}
```

UIEvent

```
- (void)drawRect:(CGRect)dirtyRect  
{  
  
}
```

UIEvent

```
- (void)drawRect:(CGRect)dirtyRect  
{  
    [[UIColor whiteColor] set];  
    UIRectFill(dirtyRect);  
}
```

UIEvent

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill(dirtyRect);

    [[UIColor redColor] set];
    [bezierPath stroke];
}
```

UIEvent

```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill(dirtyRect);

    [[UIColor redColor] set];
    [bezierPath stroke];

    CGRect ovalRect = CGRectMake(
        [bezierPath currentPoint].x-3,
        [bezierPath currentPoint].y-3, 6, 6);
}
```

UIEvent

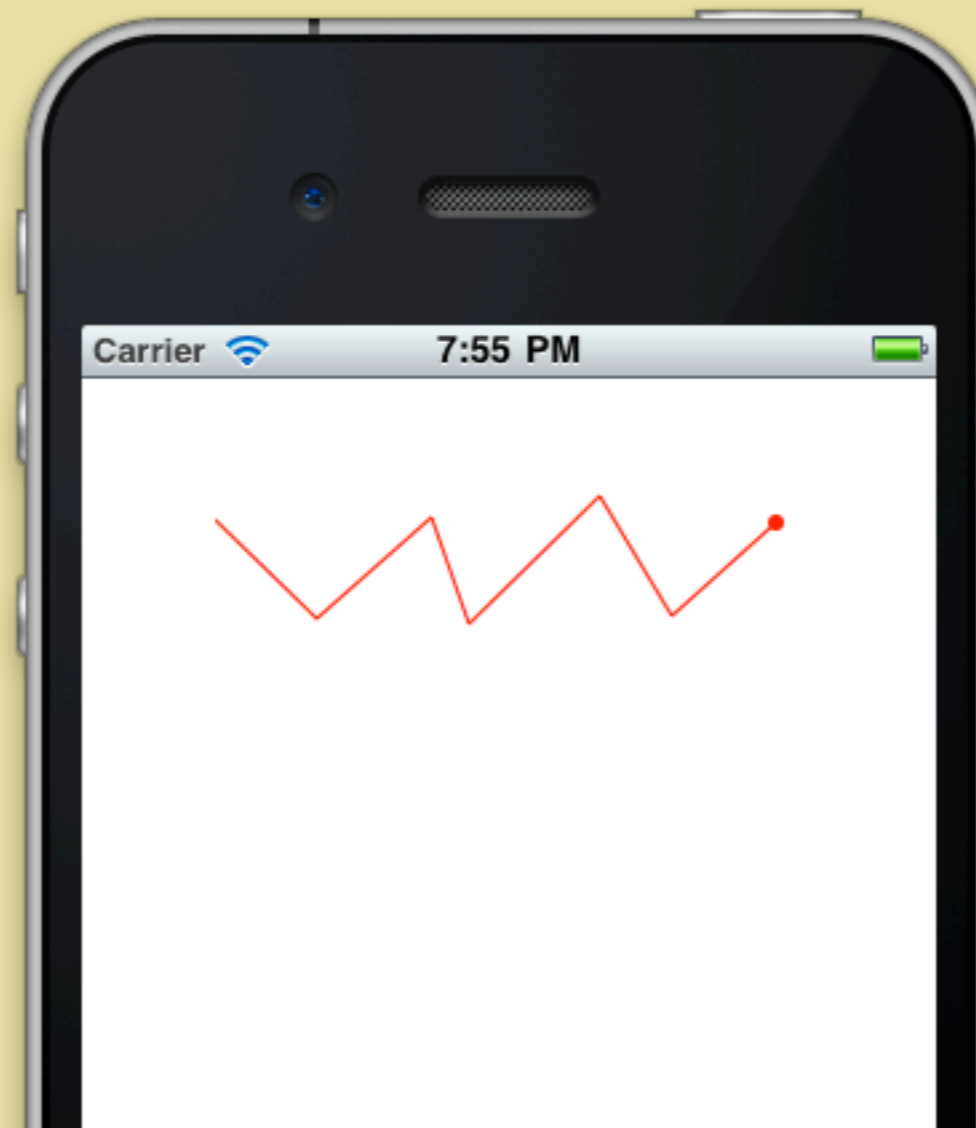
```
- (void)drawRect:(CGRect)dirtyRect
{
    [[UIColor whiteColor] set];
    UIRectFill(dirtyRect);

    [[UIColor redColor] set];
    [bezierPath stroke];

    CGRect ovalRect = CGRectMake(
        [bezierPath currentPoint].x-3,
        [bezierPath currentPoint].y-3, 6, 6);

    [[UIBezierPath bezierPathWithOvalInRect:ovalRect] fill];
}
```

UIEvent



Threading

Threading

On the Main Thread

Questions?

I'll be at the pub