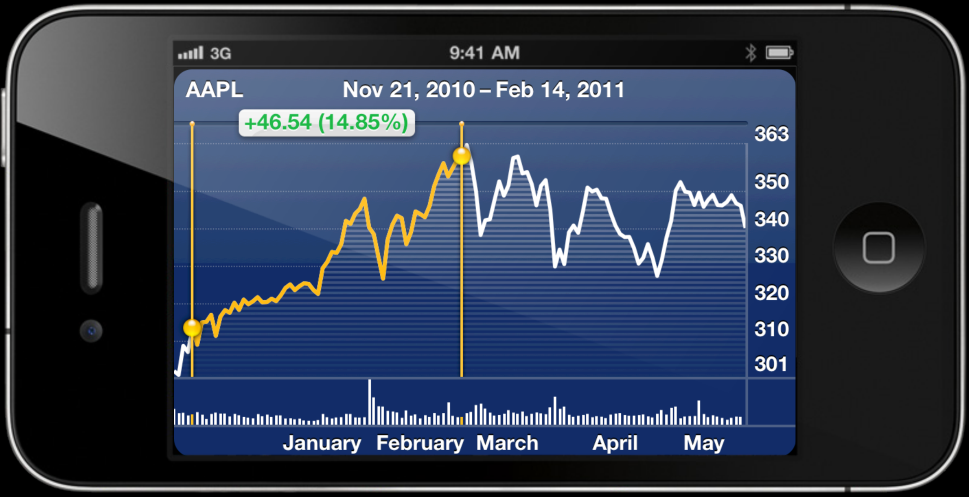
Practical Drawing for iOS Developers

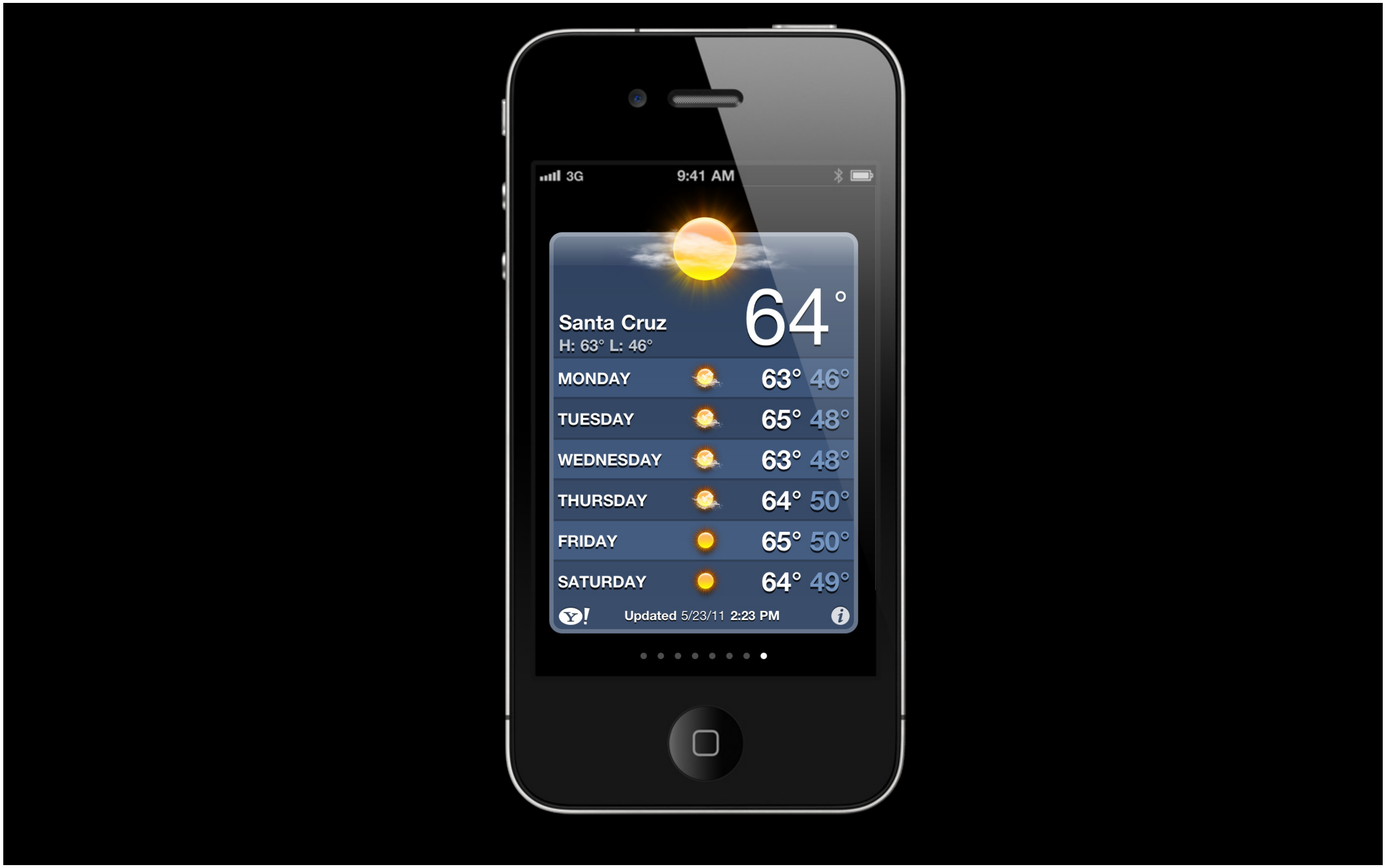
Taking advantage of the Core Graphics API

Session 129

Bill Dudney
Secret Service Captain or whatever…

What?! I could draw this with Quartz?!


3G
9:41 AM
AAPL
Nov 21, 2010 – Feb 14, 2011
+46.54 (14.85%)
363
350
340
330
320
310
301
January
February
March
April
May

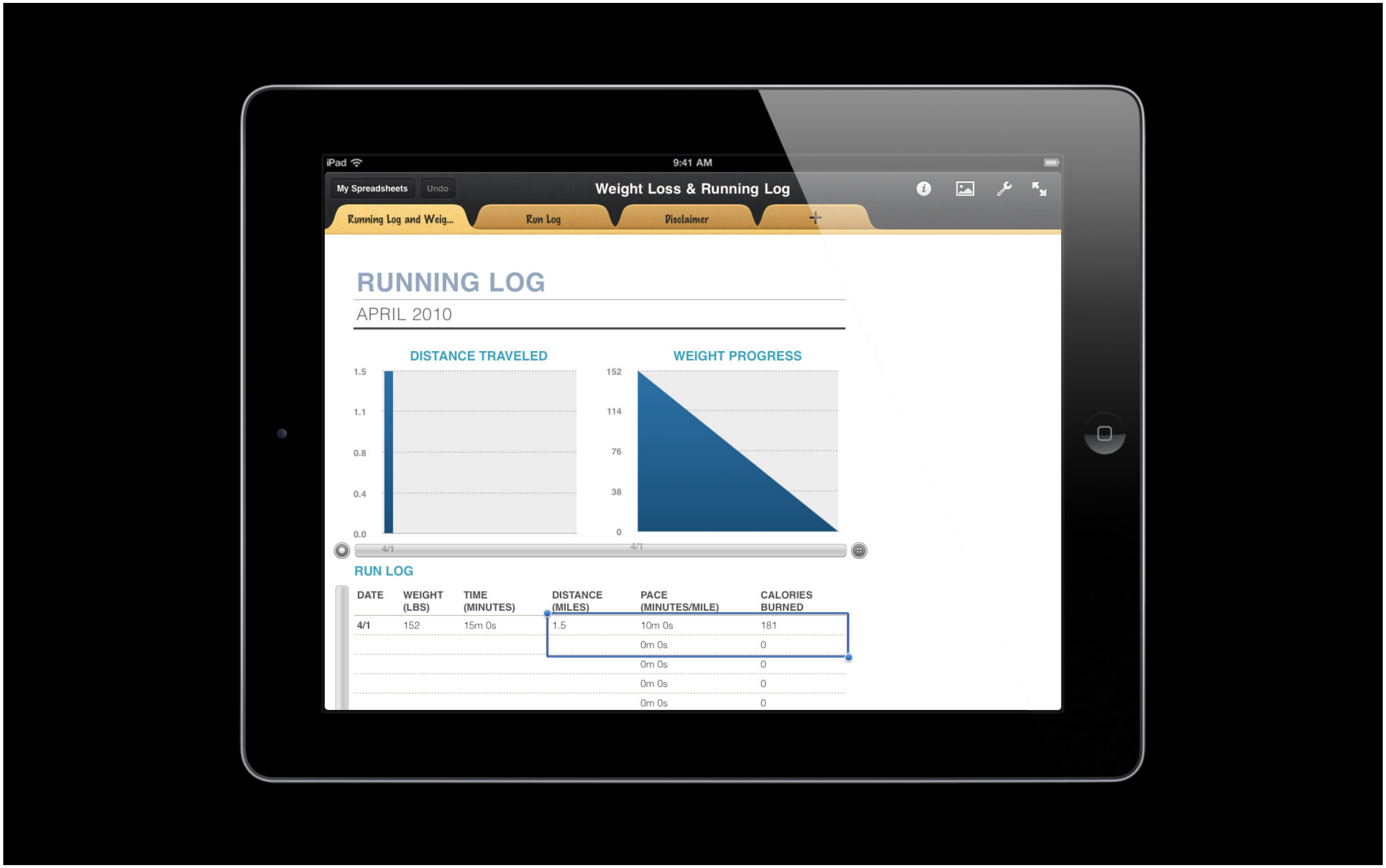


3G 9:41 AM

Santa Cruz
H: 63° L: 46°
64°

MONDAY	63°	46°
TUESDAY	65°	48°
WEDNESDAY	63°	48°
THURSDAY	64°	50°
FRIDAY	65°	50°
SATURDAY	64°	49°

Updated 5/23/11 2:23 PM



RUNNING LOG

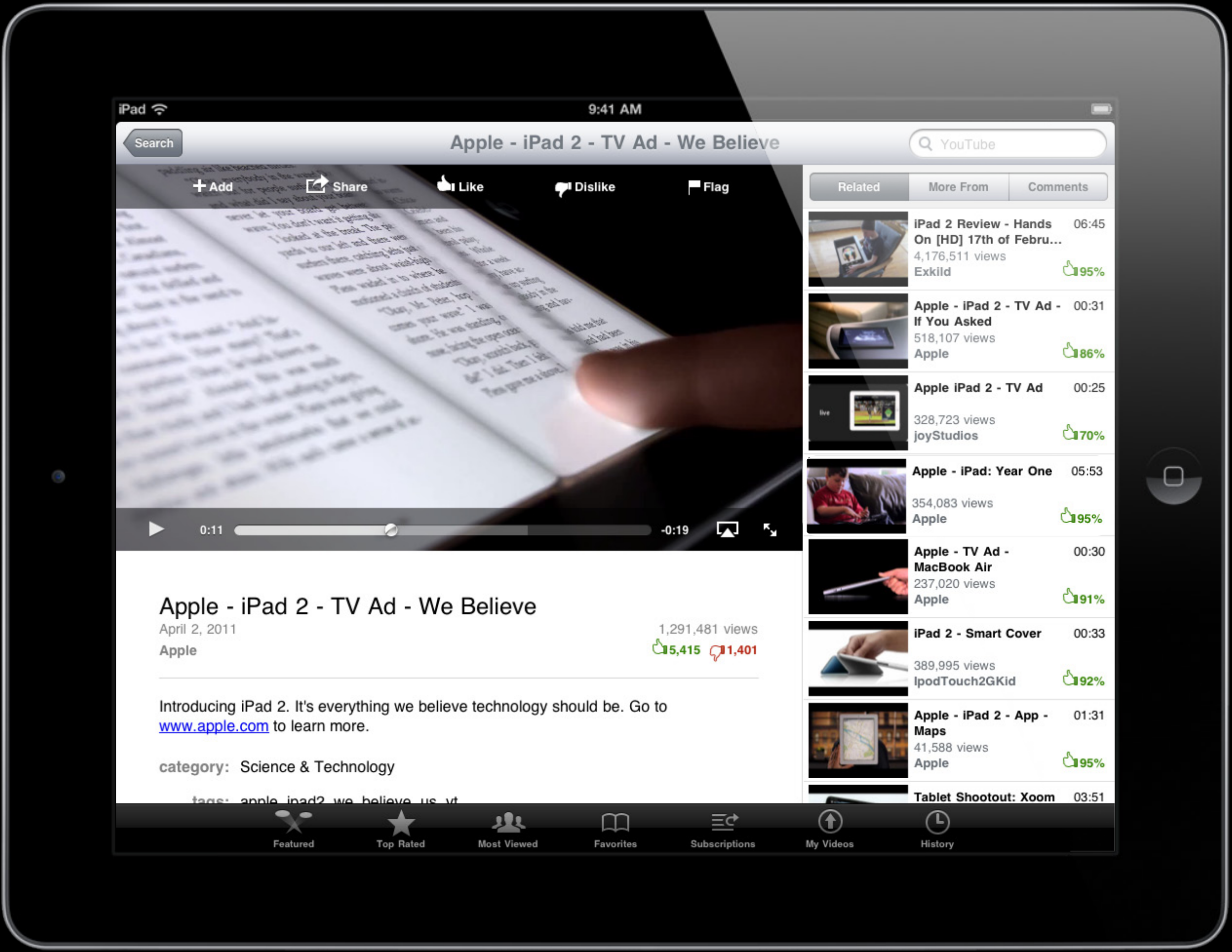
APRIL 2010

DISTANCE TRAVELED

WEIGHT PROGRESS

RUN LOG

DATE	WEIGHT (LBS)	TIME (MINUTES)	DISTANCE (MILES)	PACE (MINUTES/MILE)	CALORIES BURNED
4/1	152	15m 0s	1.5	10m 0s	181
				0m 0s	0
				0m 0s	0
				0m 0s	0
				0m 0s	0



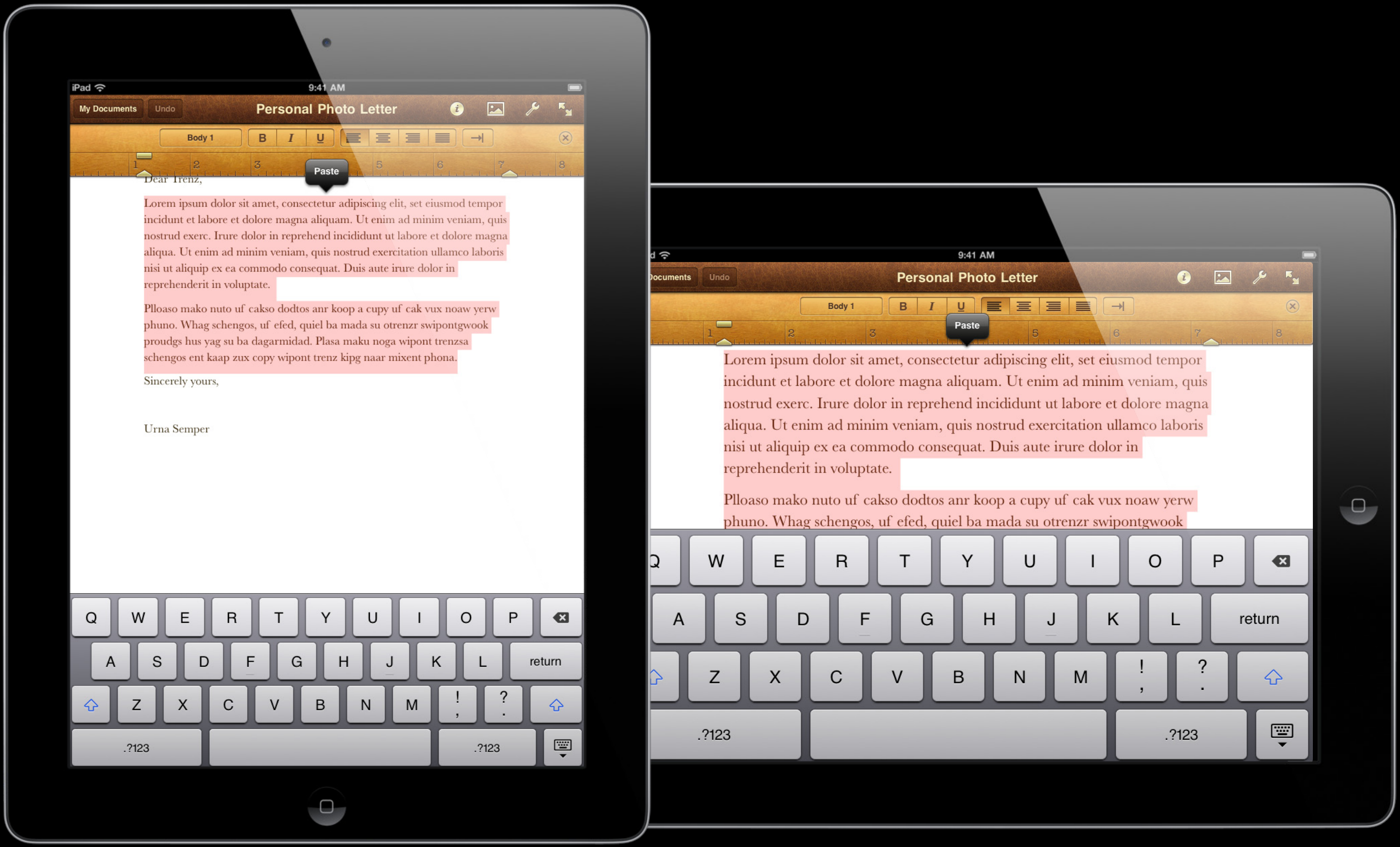
Apple - iPad 2 - TV Ad - We Believe
April 2, 2011
Apple
1,291,481 views
5,415 1,401

Introducing iPad 2. It's everything we believe technology should be. Go to www.apple.com to learn more.

category: Science & Technology

tags: apple ipad2 we believe us yt

iPad 2 Review - Hands On [HD] 17th of Febru...	06:45	4,176,511 views	Exkild	95%
Apple - iPad 2 - TV Ad - If You Asked	00:31	518,107 views	Apple	86%
Apple iPad 2 - TV Ad	00:25	328,723 views	joyStudios	70%
Apple - iPad: Year One	05:53	354,083 views	Apple	95%
Apple - TV Ad - MacBook Air	00:30	237,020 views	Apple	91%
iPad 2 - Smart Cover	00:33	389,995 views	IpodTouch2GKid	92%
Apple - iPad 2 - App - Maps	01:31	41,588 views	Apple	95%
Tablet Shootout: Xoom	03:51			



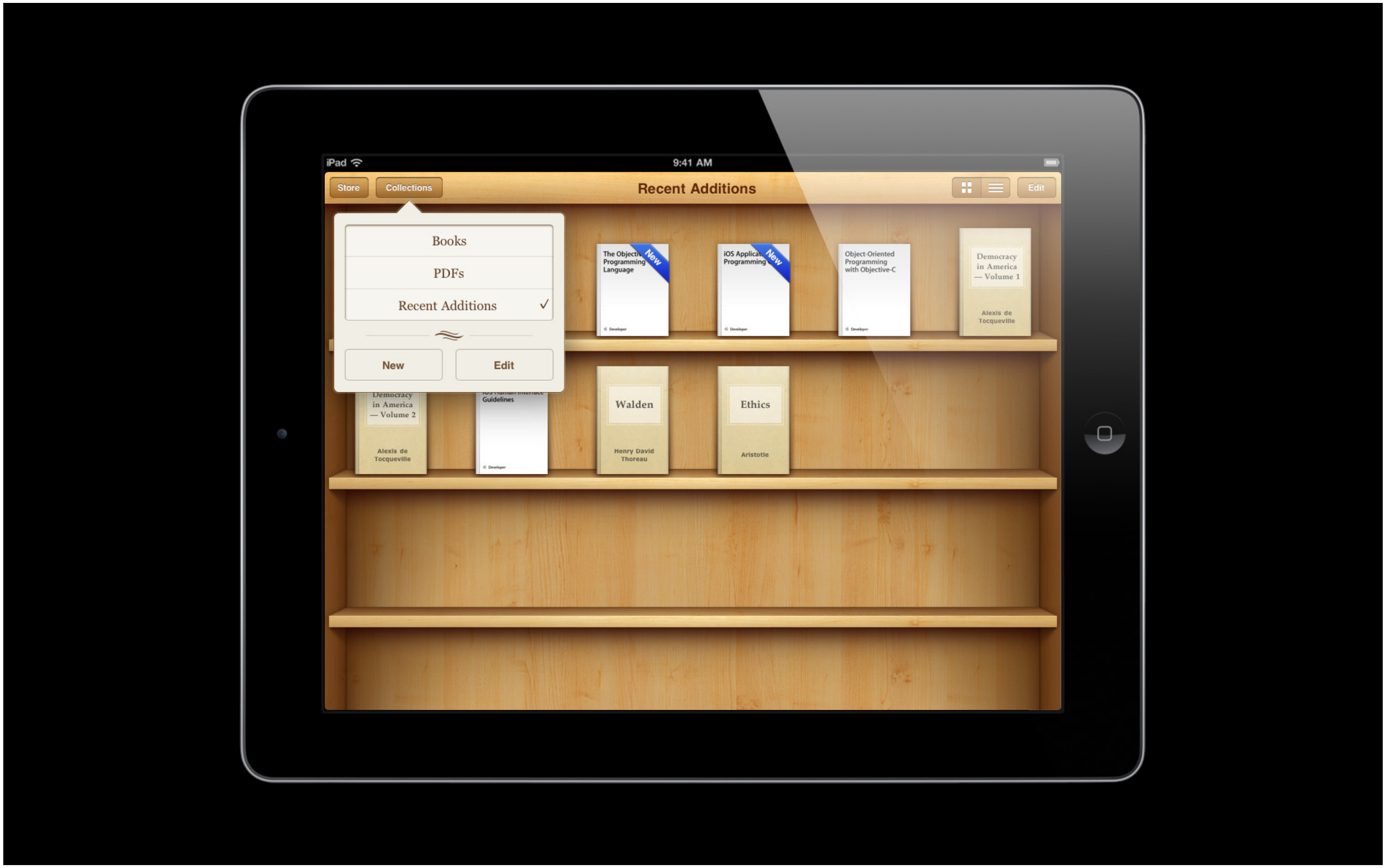
Dear Trenz,

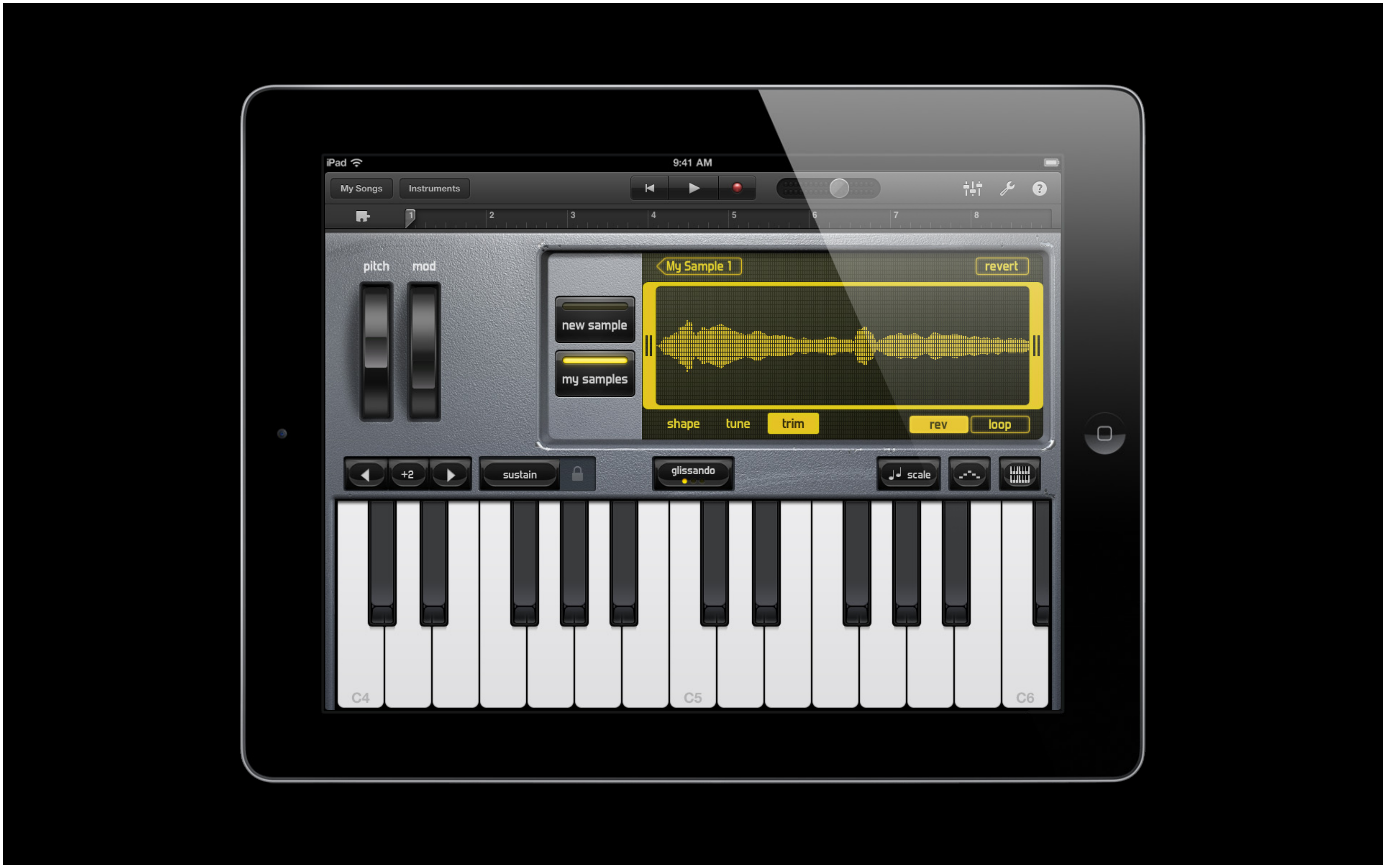
Lorem ipsum dolor sit amet, consectetur adipiscing elit, set eiusmod tempor incidunt et labore et dolore magna aliquam. Ut enim ad minim veniam, quis nostrud exerc. Irure dolor in reprehend incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate.

Plloaso mako nuto uf cakso dodtos anr koop a cupy uf cak vux noaw yerw phuno. Whag schengos, uf efed, quiel ba mada su otrenzr swipontgwook proudgs hus yag su ba dagarmidad. Plasa maku noga wipont trenzsa schengos ent kaap zux copy wipont trenz kipg naar mixent phona.

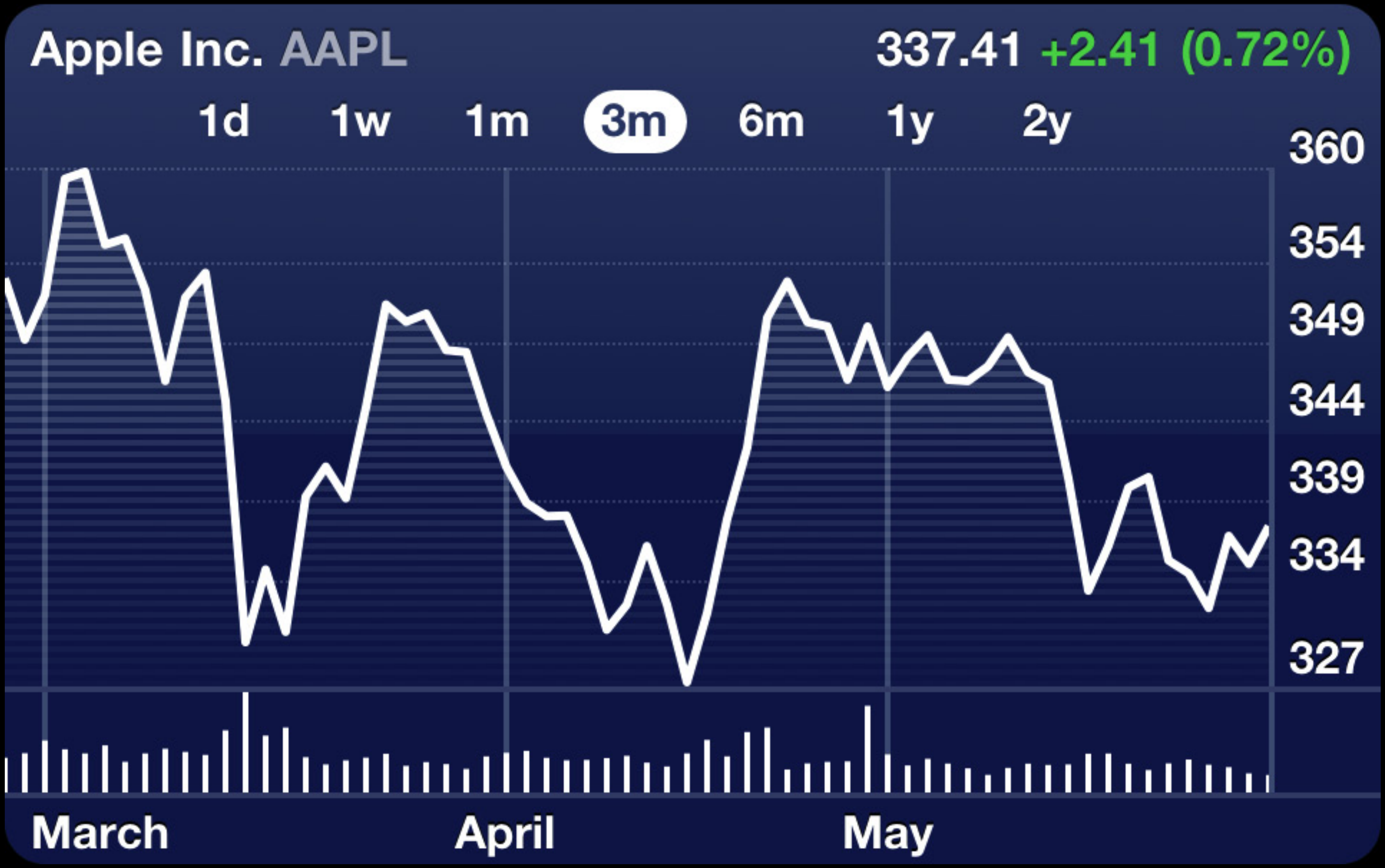
Sincerely yours,

Urna Semper


iPad
9:41 AM
Store
Collections
Recent Additions
Edit
Books
PDFs
Recent Additions ✓
New
Edit
The Objective-C Programming Language
iOS Application Programming
Object-Oriented Programming with Objective-C
Democracy in America — Volume 1
Alexis de Tocqueville
Democracy in America — Volume 2
Alexis de Tocqueville
iOS Human Interface Guidelines
Walden
Henry David Thoreau
Ethics
Aristotle


iPad
9:41 AM
My Songs
Instruments
pitch
mod
new sample
my samples
My Sample 1
revert
shape
tune
trim
rev
loop
+2
sustain
glissando
scale
C4
C5
C6

Agenda

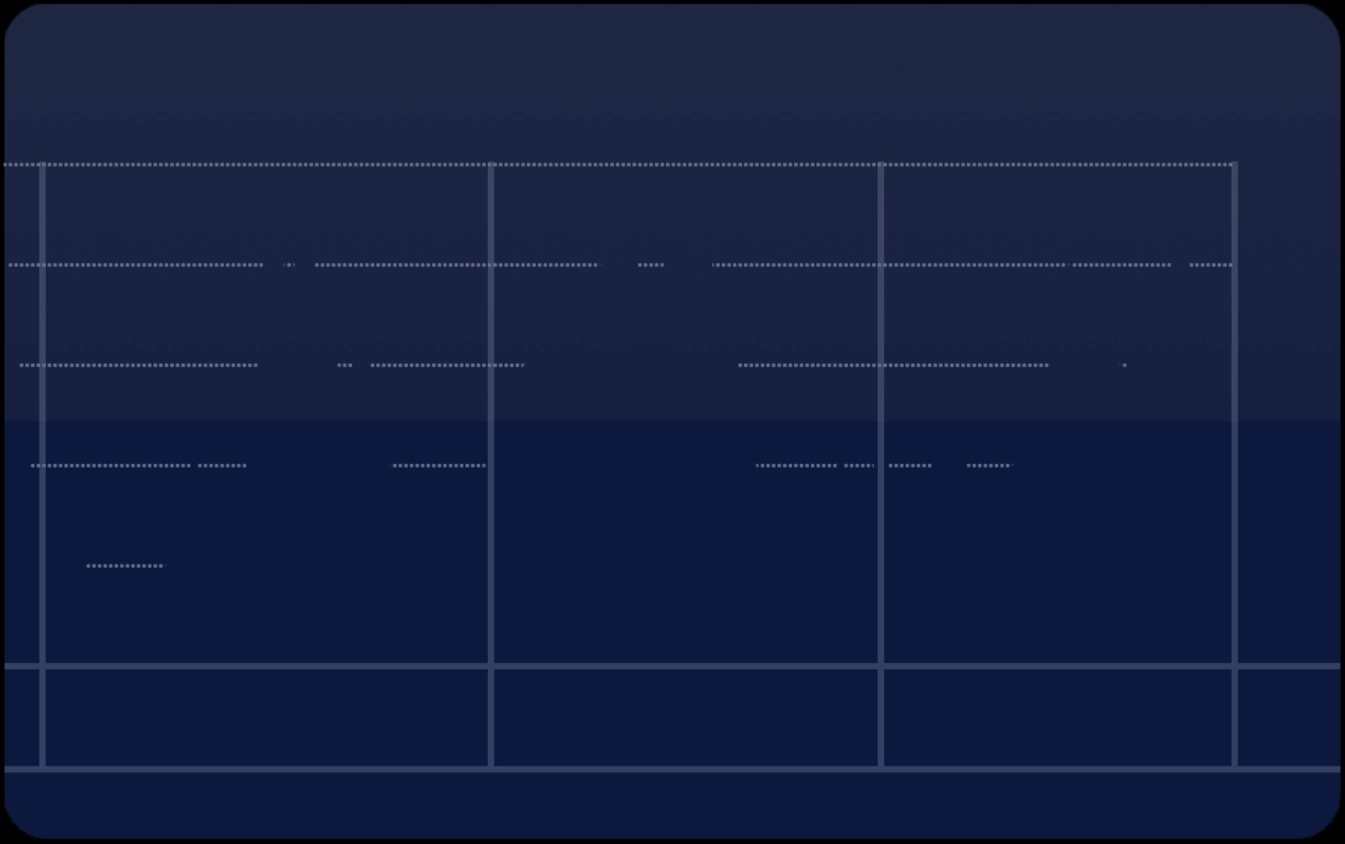

Apple Inc. AAPL
337.41 +2.41 (0.72%)
1d
1w
1m
3m
6m
1y
2y
360
354
349
344
339
334
327
March
April
May

Gradient Background

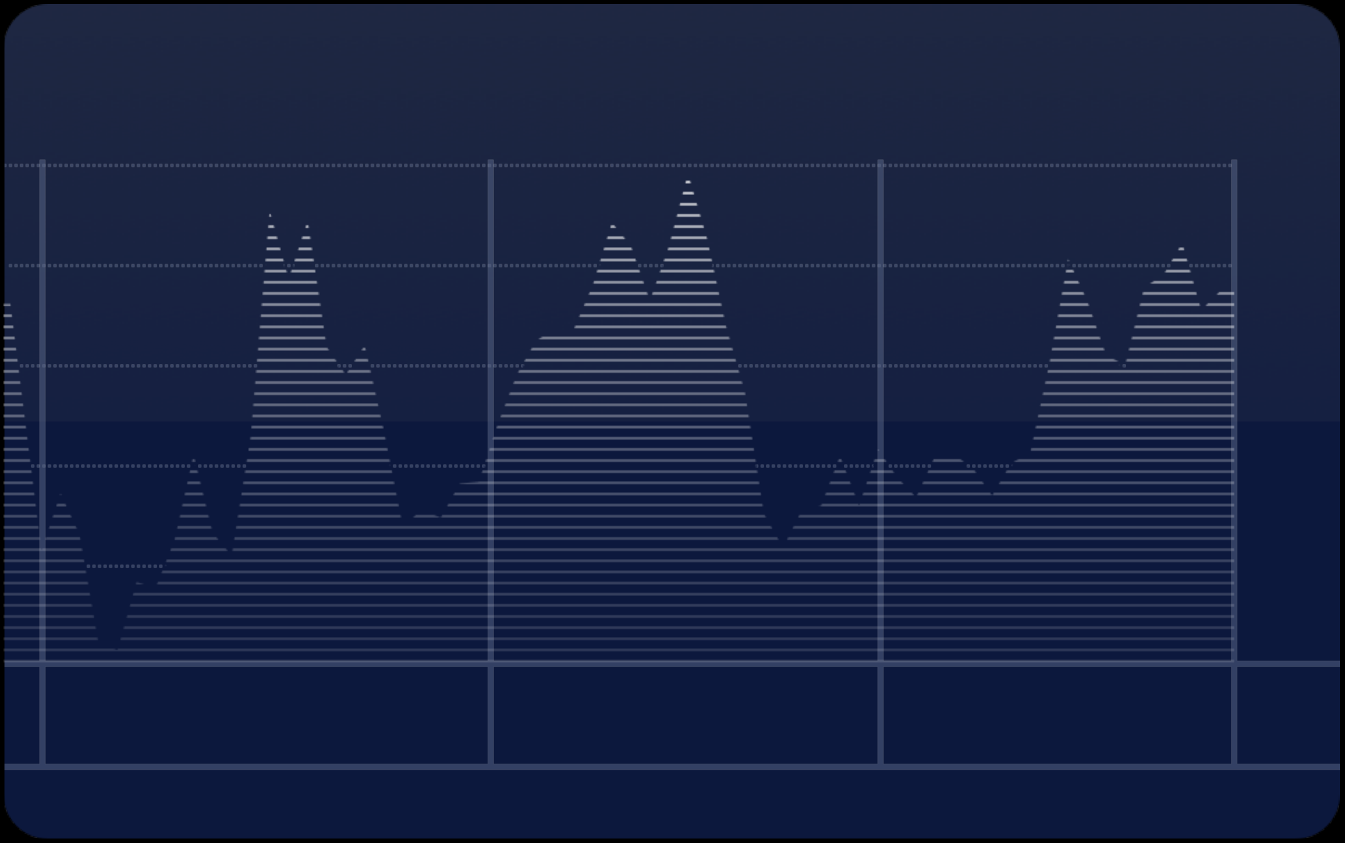
Gradient Background Clipped

Data Grid

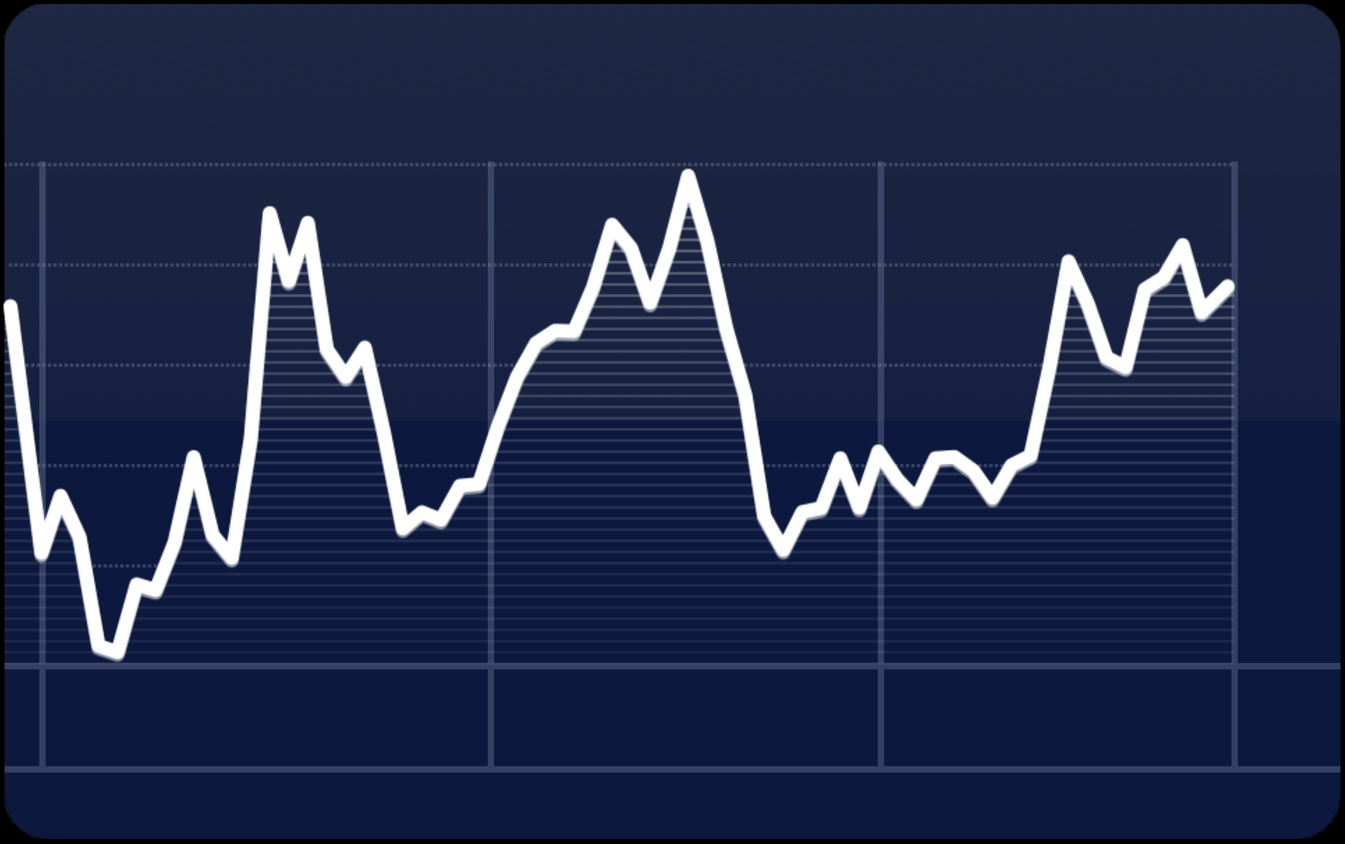
Clipped Horizontal Grid



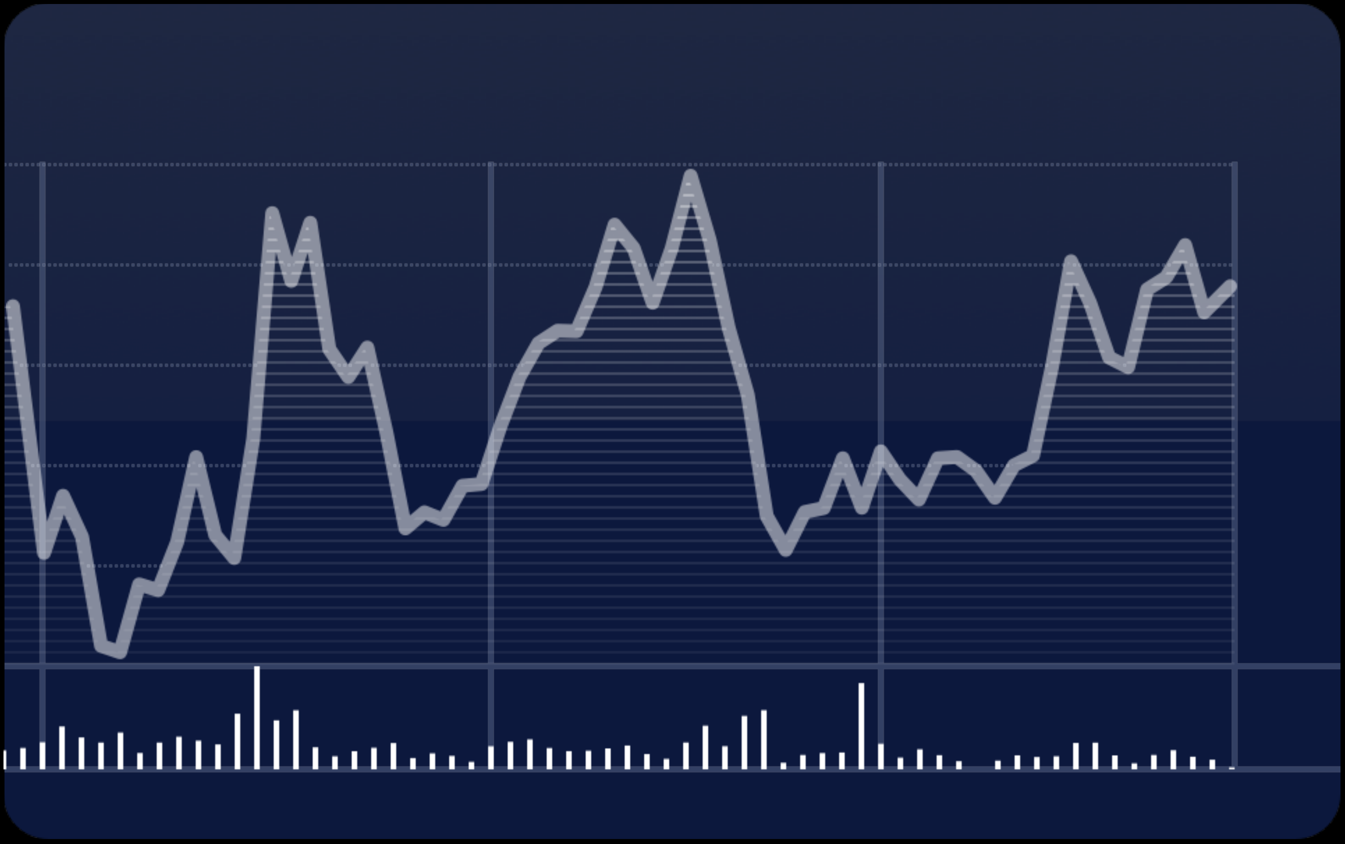
Clipped Linear Fill



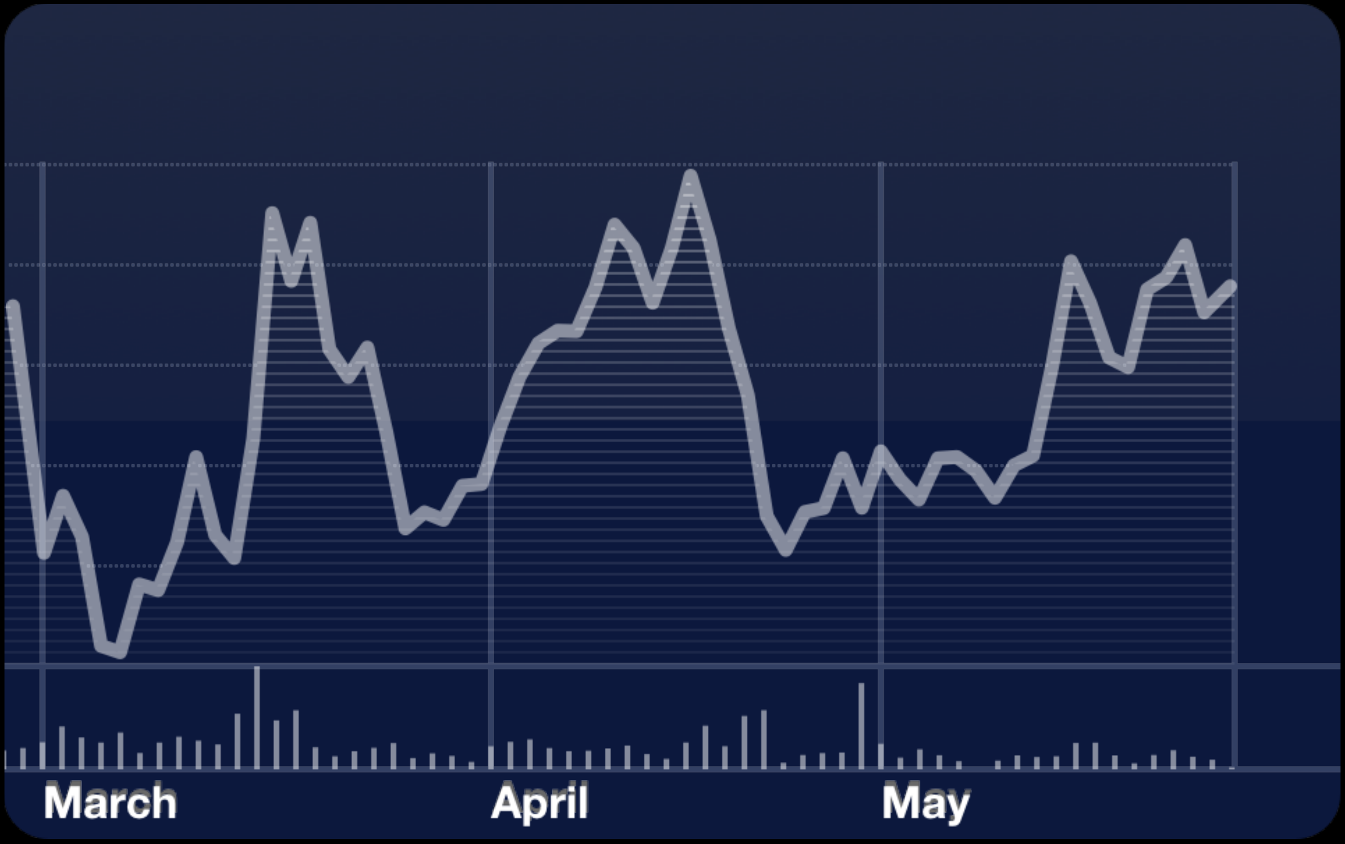
Closing Data



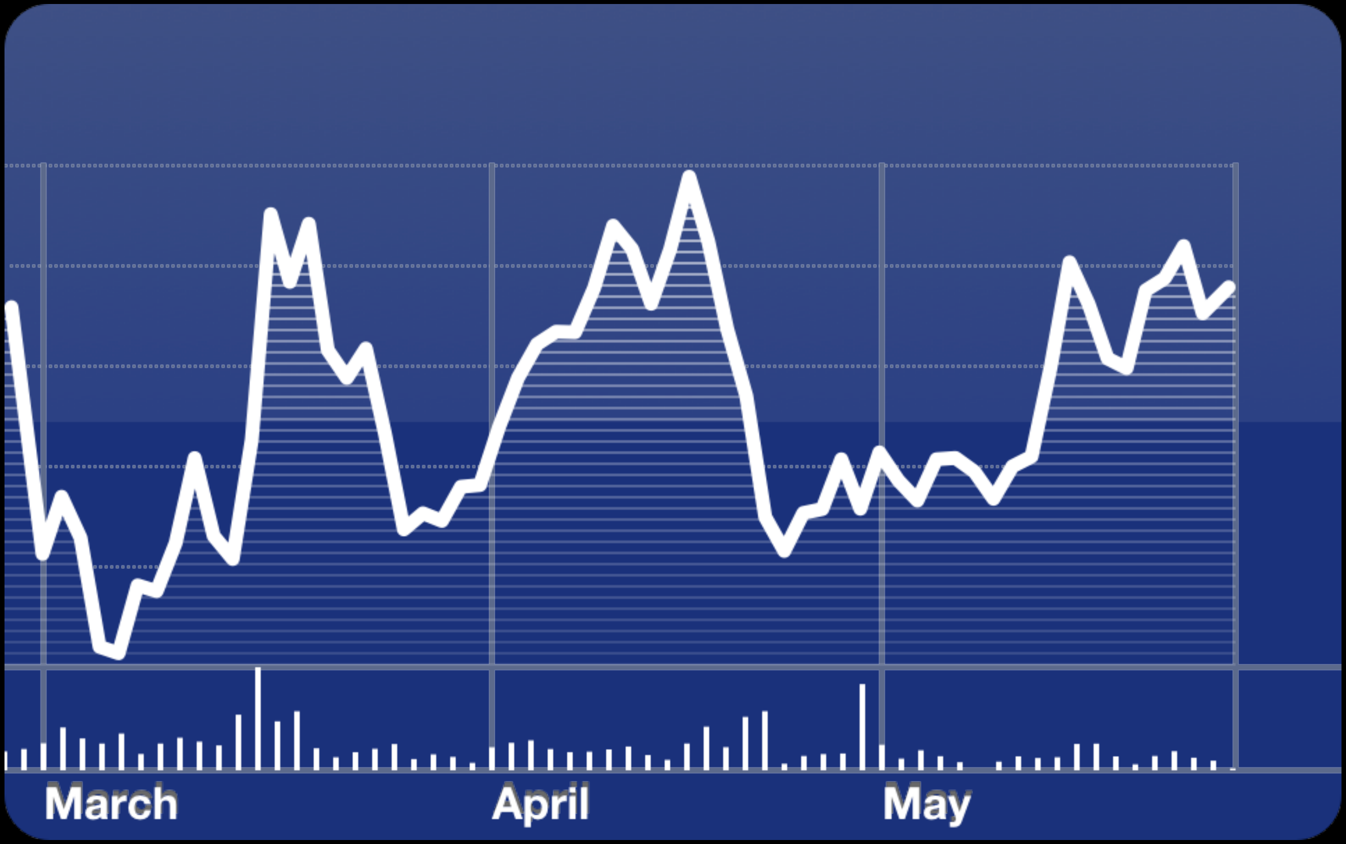
Volume Data



Text Labels



Simple Stocks



Drawing

Color Fill

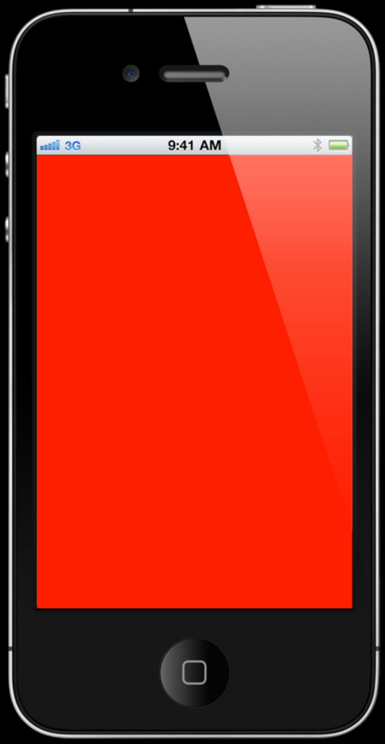
```
@implementation MyView
...
- (void)drawRect:(CGRect)rect {
    ...
}
...
@end
```

Color Fill

```
@implementation MyView
...
- (void)drawRect:(CGRect)rect {
    [[UIColor redColor] setFill];
    UIRectFill(self.bounds);
}
...
@end
```

Color Fill

```
@implementation MyView
...
- (void)drawRect:(CGRect)rect {
    [[UIColor redColor] setFill];
    UIRectFill(self.bounds);
}
...
@end
```



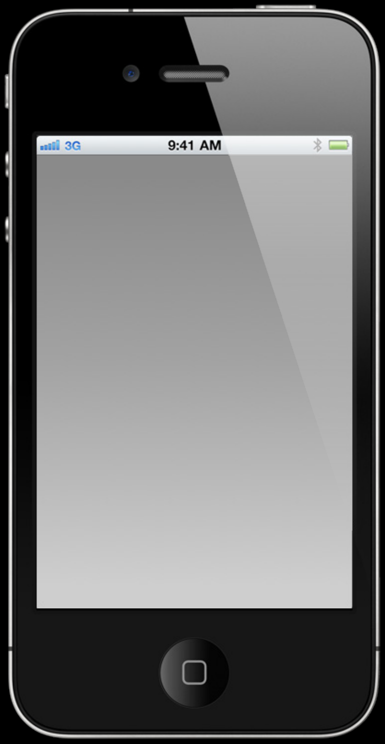
Gradient Fill

```
- (void)drawRect:(CGRect)rect {
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGGradientRef gradient = [self gradient];

    CGPoint startPoint =
            CGPointMake(CGRectGetMidX(self.bounds), 0.0);
    CGPoint endPoint =
            CGPointMake(CGRectGetMidX(self.bounds),
                        CGRectGetMaxY(self.bounds));

    CGContextDrawLinearGradient(ctx, gradient,
                                startPoint, endPoint, 0);
}
```



Core Graphics Is a C API

Quartz 2D Documentation

DON'T PANIC

…just now. —Bill Dudney

UIKit to the Rescue

Much of Core Graphics is covered by UIKit

Gradient Fill

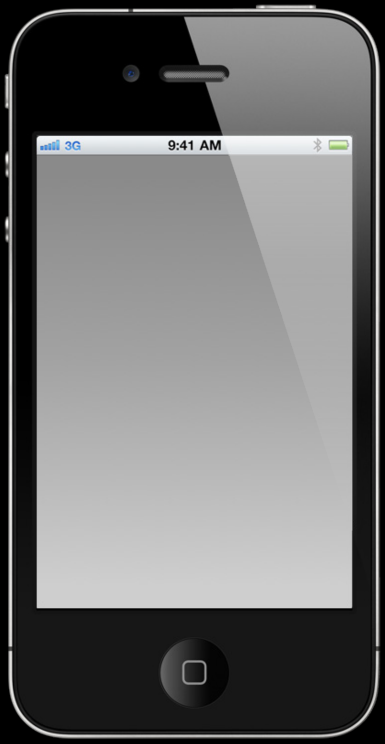
Get the context

```
- (void)drawRect:(CGRect)rect {
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGGradientRef gradient = [self gradient];

    CGPoint startPoint =
            CGPointMake(CGRectGetMidX(self.bounds), 0.0);
    CGPoint endPoint =
            CGPointMake(CGRectGetMidX(self.bounds),
                        CGRectGetMaxY(self.bounds));

    CGContextDrawLinearGradient(ctx, gradient,
                                startPoint, endPoint, 0);
}
```



Gradient Fill

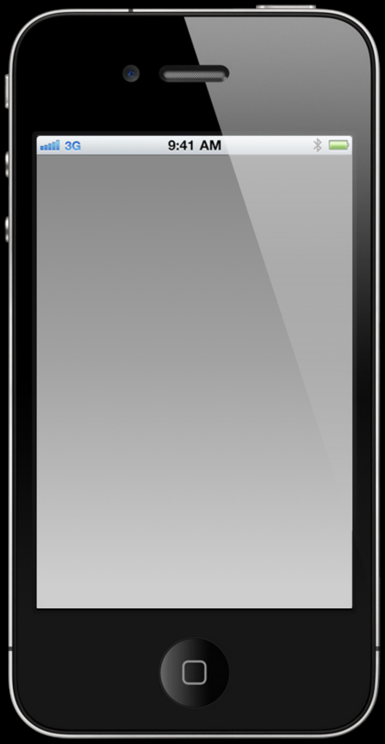
Get the gradient

```
- (void)drawRect:(CGRect)rect {
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGGradientRef gradient = [self gradient];

    CGPoint startPoint =
            CGPointMake(CGRectGetMidX(self.bounds), 0.0);
    CGPoint endPoint =
            CGPointMake(CGRectGetMidX(self.bounds),
                        CGRectGetMaxY(self.bounds));

    CGContextDrawLinearGradient(ctx, gradient,
                            startPoint, endPoint, 0);
}
```



Gradient Fill

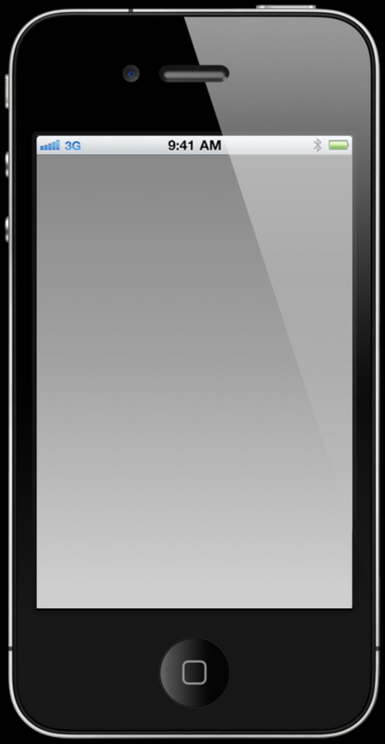
Create start and end points

```
- (void)drawRect:(CGRect)rect {
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGGradientRef gradient = [self gradient];

    CGPoint startPoint =
            CGPointMake(CGRectGetMidX(self.bounds), 0.0);
    CGPoint endPoint =
            CGPointMake(CGRectGetMidX(self.bounds),
                        CGRectGetMaxY(self.bounds));

    CGContextDrawLinearGradient(ctx, gradient,
                                startPoint, endPoint, 0);
}
```



Gradient Fill

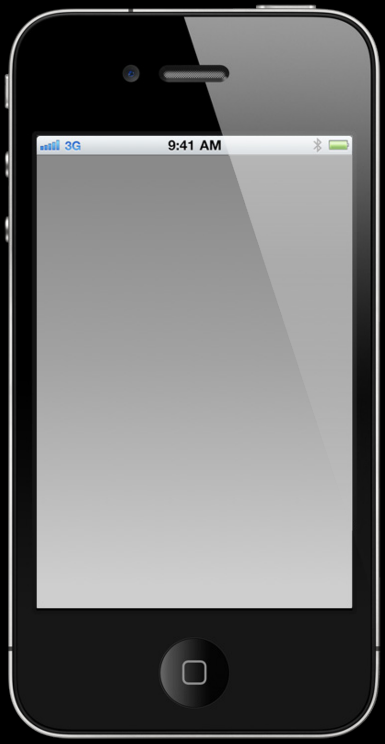
Draw the gradient

```
- (void)drawRect:(CGRect)rect {
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGGradientRef gradient = [self gradient];

    CGPoint startPoint =
            CGPointMake(CGRectGetMidX(self.bounds), 0.0);
    CGPoint endPoint =
            CGPointMake(CGRectGetMidX(self.bounds),
                        CGRectGetMaxY(self.bounds));

    CGContextDrawLinearGradient(ctx, gradient,
                                startPoint, endPoint, 0);
}
```



Create the Gradient

The colors

```
- (CGGradientRef)gradient {
    if(NULL == _gradient) {
        CGFloat colors[6] = {138.0f/255.0f, 1.0f,
                             162.0f/255.0f, 1.0f,
                             206.0f/255.0f, 1.0f};
        CGFloat locations[3] = {0.05f, 0.45f, 0.95f};

        CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceGray();
        _gradient = CGGradientCreateWithColorComponents(colorSpace, colors,
                                                        locations, 3);
        CGColorSpaceRelease(colorSpace);
    }
    return _gradient;
}
```

Create the Gradient

The color stops

```
- (CGGradientRef)gradient {
    if(NULL == _gradient) {
        CGFloat colors[6] = {138.0f/255.0f, 1.0f,
                             162.0f/255.0f, 1.0f,
                             206.0f/255.0f, 1.0f};
        CGFloat locations[3] = {0.05f, 0.45f, 0.95f};

        CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceGray();
        _gradient = CGGradientCreateWithColorComponents(colorSpace, colors,
                                                        locations, 3);
        CGColorSpaceRelease(colorSpace);
    }
    return _gradient;
}
```

Create the Gradient

The color space

```
- (CGGradientRef)gradient {
    if(NULL == _gradient) {
        CGFloat colors[6] = {138.0f/255.0f, 1.0f,
                             162.0f/255.0f, 1.0f,
                             206.0f/255.0f, 1.0f};
        CGFloat locations[3] = {0.05f, 0.45f, 0.95f};

        CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceGray();
        _gradient = CGGradientCreateWithColorComponents(colorSpace, colors,
                                                        locations, 3);
        CGColorSpaceRelease(colorSpace);
    }
    return _gradient;
}
```

Create the Gradient

```
- (CGGradientRef)gradient {
    if(NULL == _gradient) {
        CGFloat colors[6] = {138.0f/255.0f, 1.0f,
                             162.0f/255.0f, 1.0f,
                             206.0f/255.0f, 1.0f};
        CGFloat locations[3] = {0.05f, 0.45f, 0.95f};

        CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceGray();
        _gradient = CGGradientCreateWithColorComponents(colorSpace, colors,
                                                        locations, 3);
        CGColorSpaceRelease(colorSpace);
    }
    return _gradient;
}
```

Create the Gradient

Cleanup

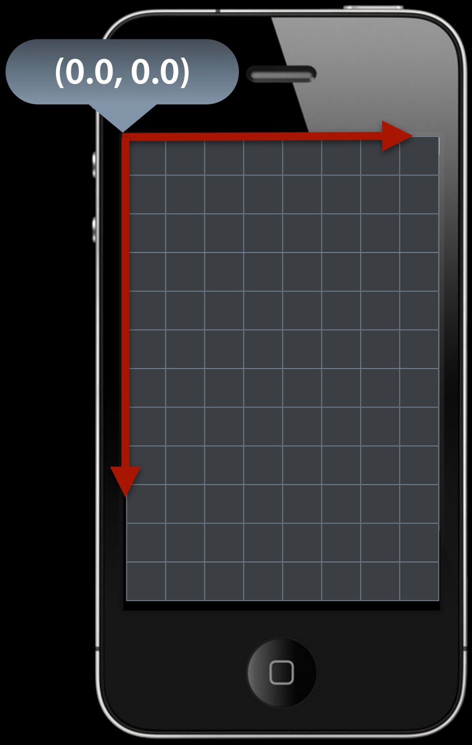
```
- (CGGradientRef)gradient {
    if(NULL == _gradient) {
        CGFloat colors[6] = {138.0f/255.0f, 1.0f,
                             162.0f/255.0f, 1.0f,
                             206.0f/255.0f, 1.0f};
        CGFloat locations[3] = {0.05f, 0.45f, 0.95f};

        CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceGray();
        _gradient = CGGradientCreateWithColorComponents(colorSpace, colors,
                                                        locations, 3);
        CGColorSpaceRelease(colorSpace);
    }
    return _gradient;
}
```

Basics

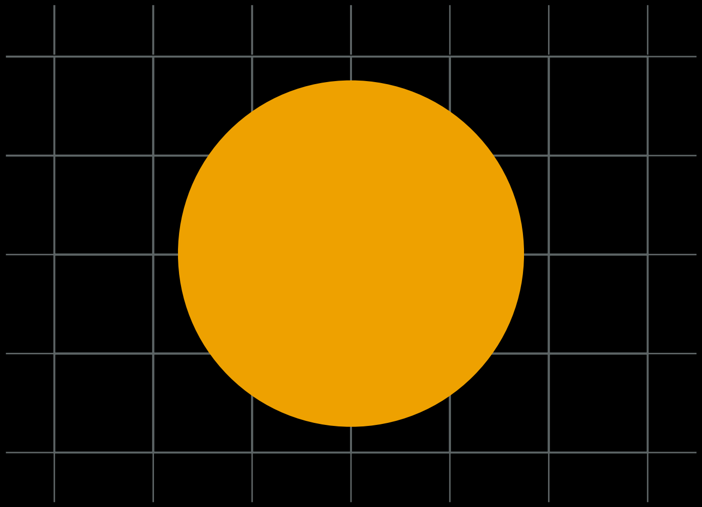
Understanding the drawing model

UIKit Coordinate System



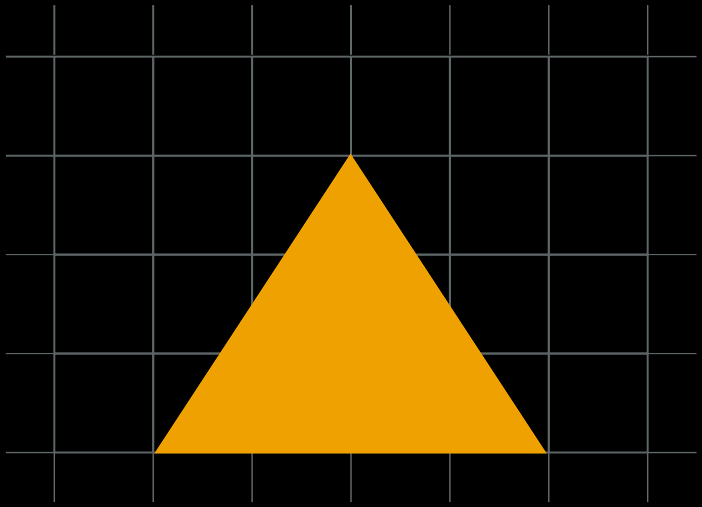
Quartz Is a Geometric System

Describe what you want drawn



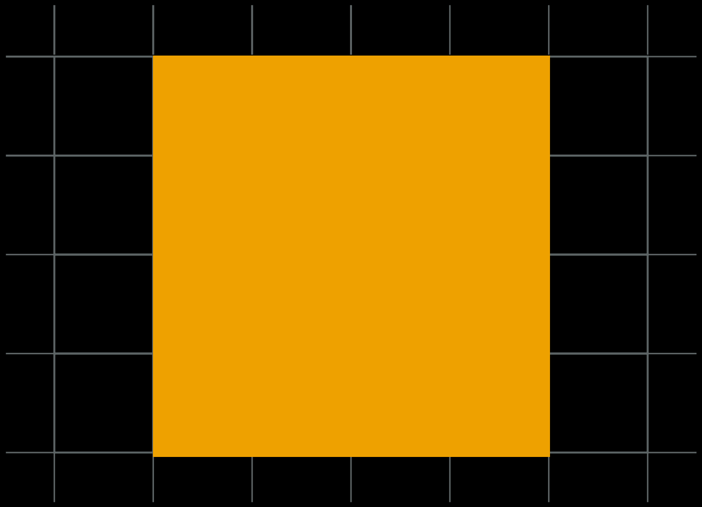
Quartz Is a Geometric System

Describe what you want drawn



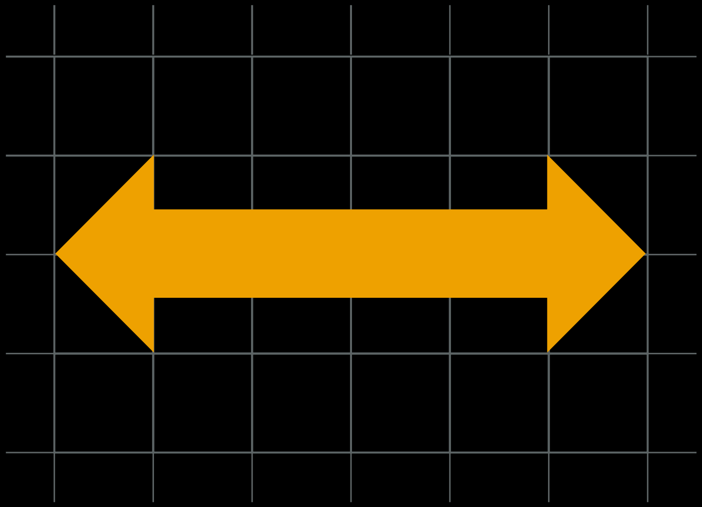
Quartz Is a Geometric System

Describe what you want drawn



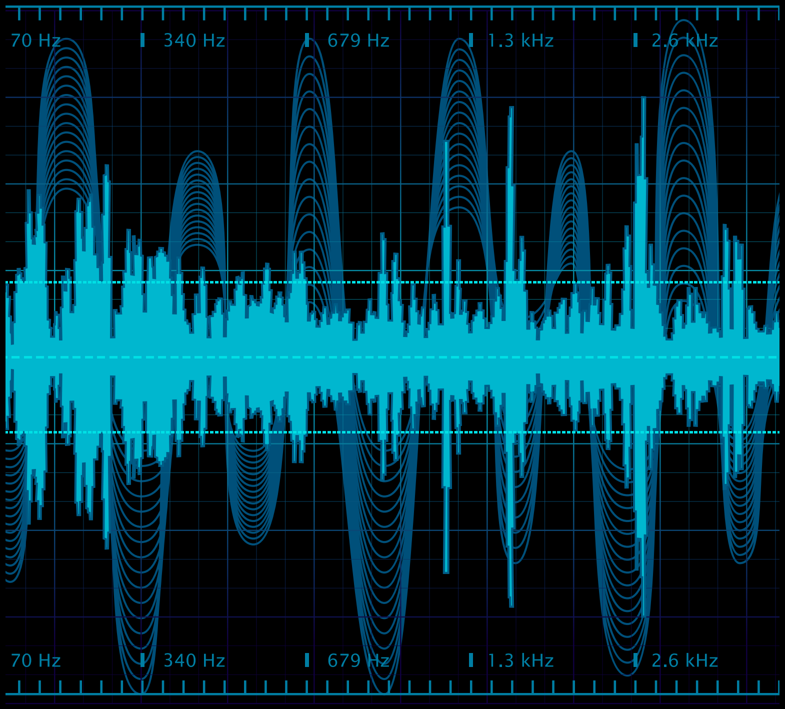
Quartz Is a Geometric System

Describe what you want drawn



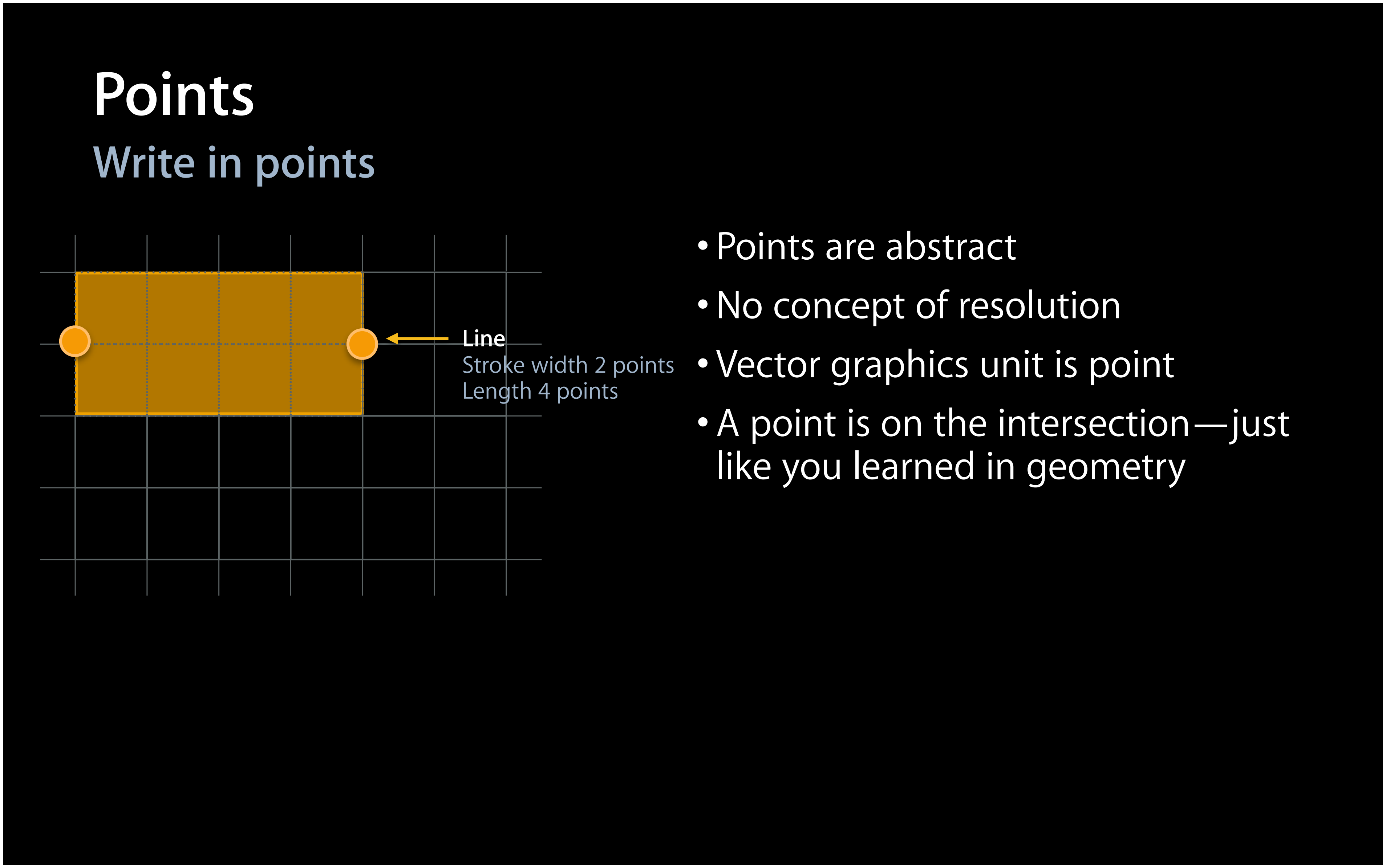
Quartz Is a Geometric System

Describe what you want drawn



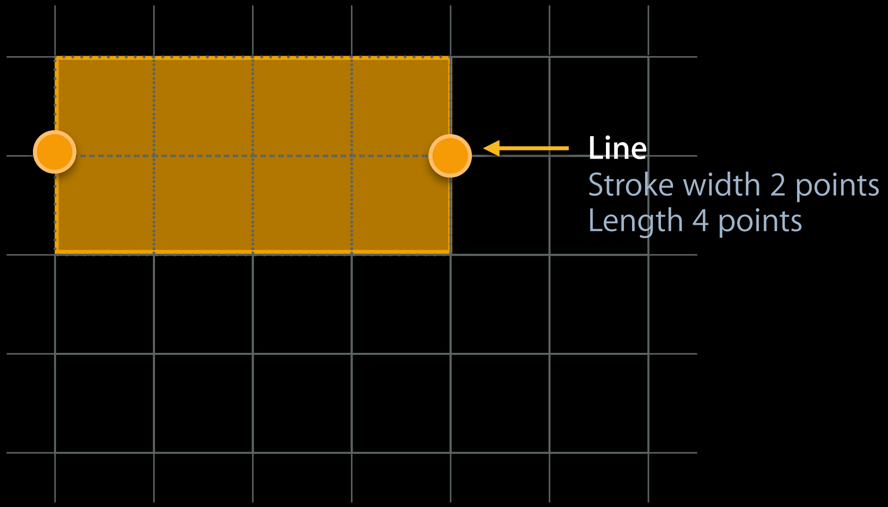
Quartz Is a Geometric System

Describe what you want drawn



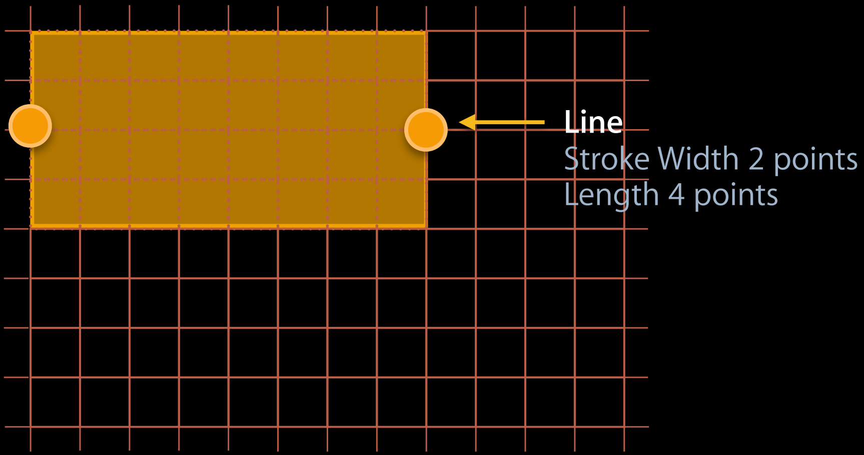
Points

Write in points

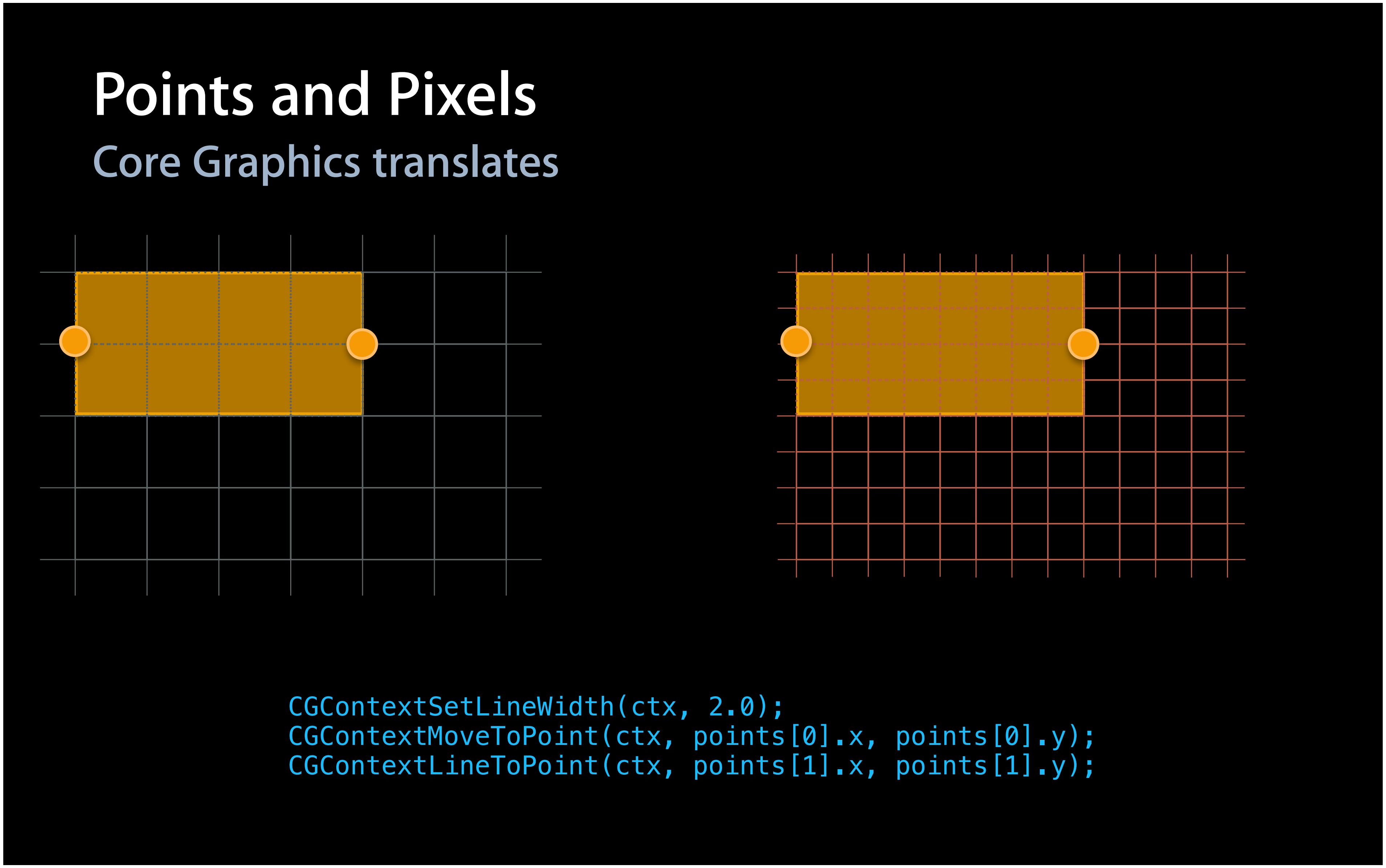


- Points are abstract
- No concept of resolution
- Vector graphics unit is point
- A point is on the intersection—just like you learned in geometry

Pixels
Hardware is pixels

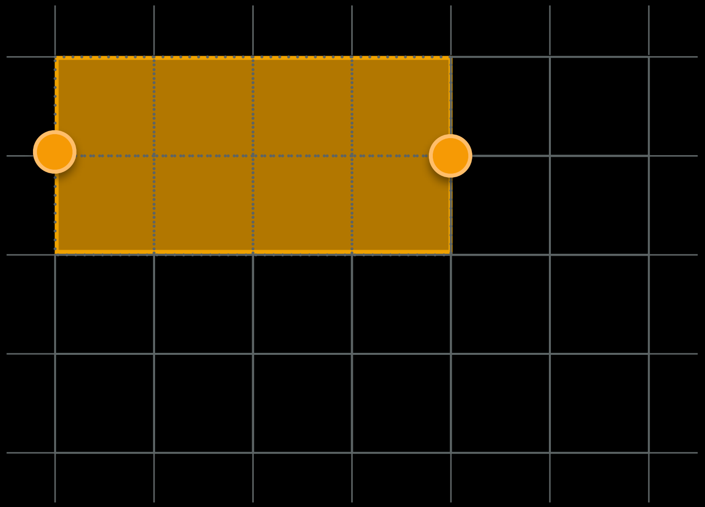


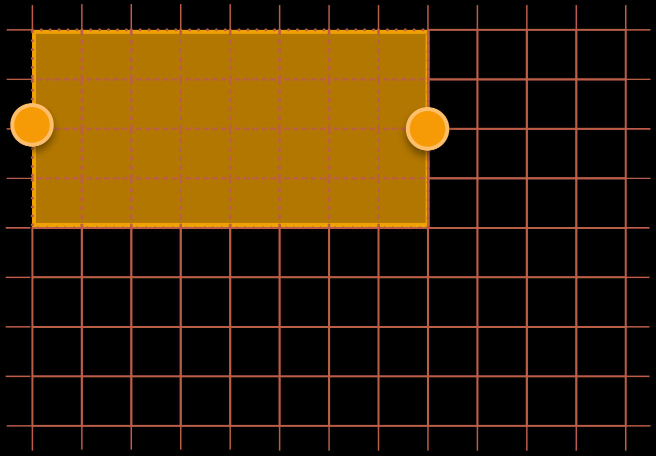
- Pixels are concrete
- One color per pixel



Points and Pixels

Core Graphics translates





```
CGContextSetLineWidth(ctx, 2.0);
CGContextMoveToPoint(ctx, points[0].x, points[0].y);
CGContextLineToPoint(ctx, points[1].x, points[1].y);
```

Thinking in Points

- 10-point system font is readable (for many eyes, anyway)
- 44x44 point rectangle is a good touch size
- 320x480 points for iPhone (3G, 3Gs, and 4) and iPod touch

CGContextRef

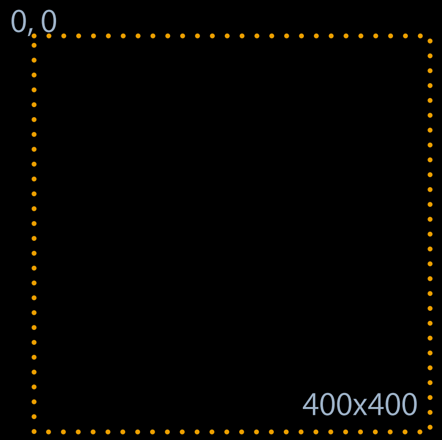
Context properties

- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix

CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix

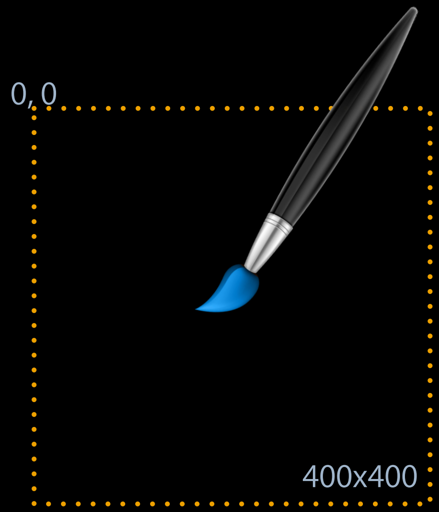


```
CGContextAddRect(ctx, CGRectMake(0.0, 0.0, 400.0, 400.0));
```

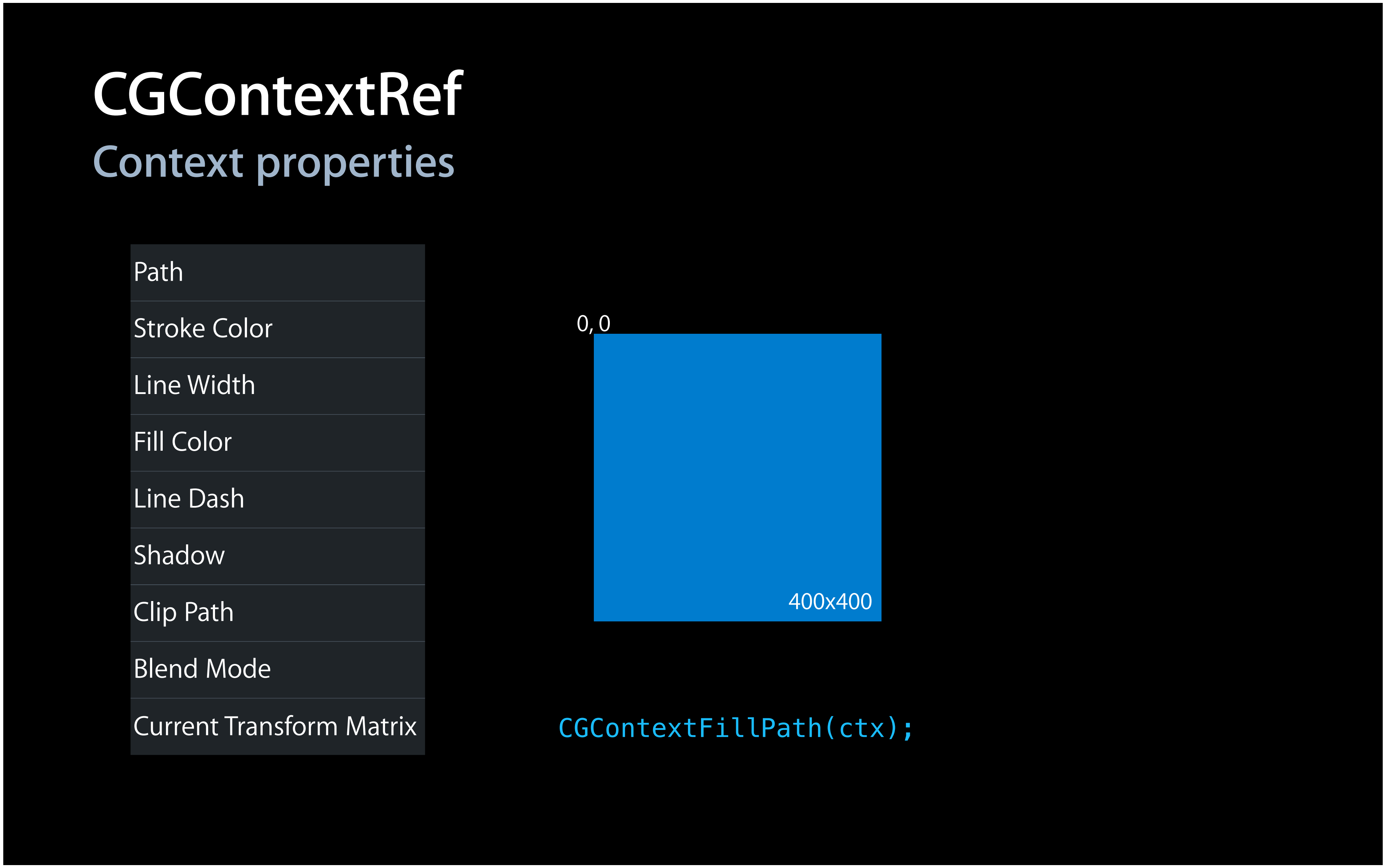
CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix



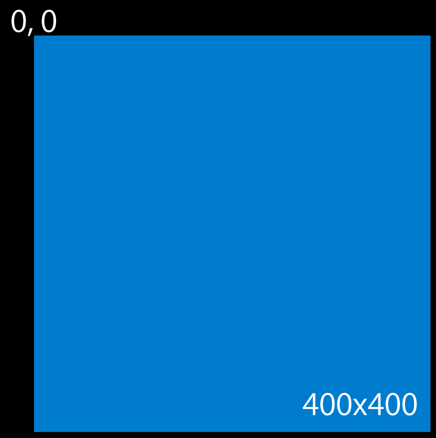
```
CGContextSetFillColorWithColor(ctx,
                       [[UIColor blueColor] CGColor]);
```



CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix

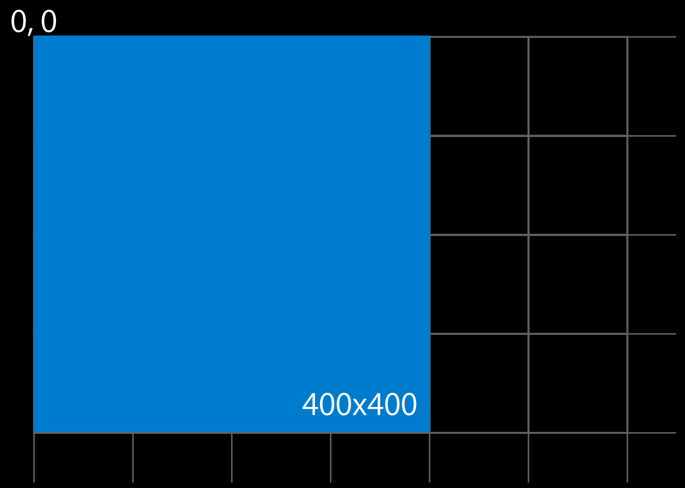


```
CGContextFillPath(ctx);
```

CGContextRef

Context properties

- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix



```
[[UIColor blueColor] setFill];
UIRectFill(CGRectMake(0.0, 0.0, 400.0, 400.0));
```

CGContextRef

Context properties

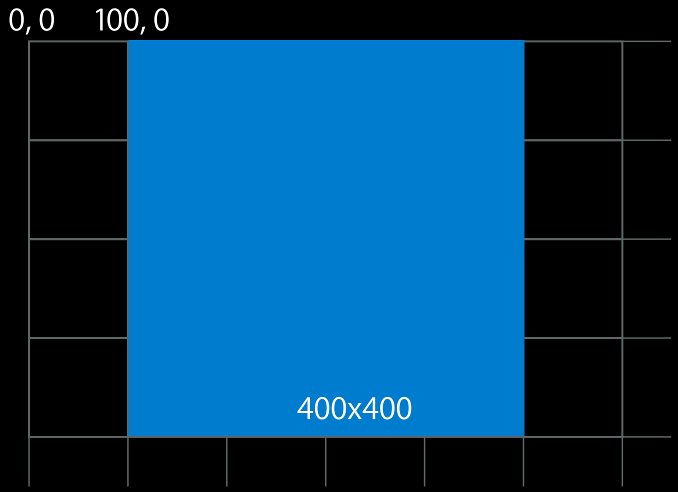
- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix

- Current transform matrix (CTM)
  - Defines “user space”
  - User space is where points live

CGContextRef

Context properties

- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix



```
CGContextTranslateCTM(ctx, 100.0, 0.0);
[[UIColor blueColor] setFill];
UIRectFill(CGRectMake(0.0, 0.0, 400.0, 400.0));
```

CGContextRef

Context properties

- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix

- Context stack
  - Save state to take a snapshot
  - Revert state to return to snapshot

```
CGContextSaveGState(ctx);
CGContextTranslateCTM(ctx, 100.0, 0.0);
// draw in transformed context
CGContextRestoreGState(ctx);
// draw in untransformed context
```

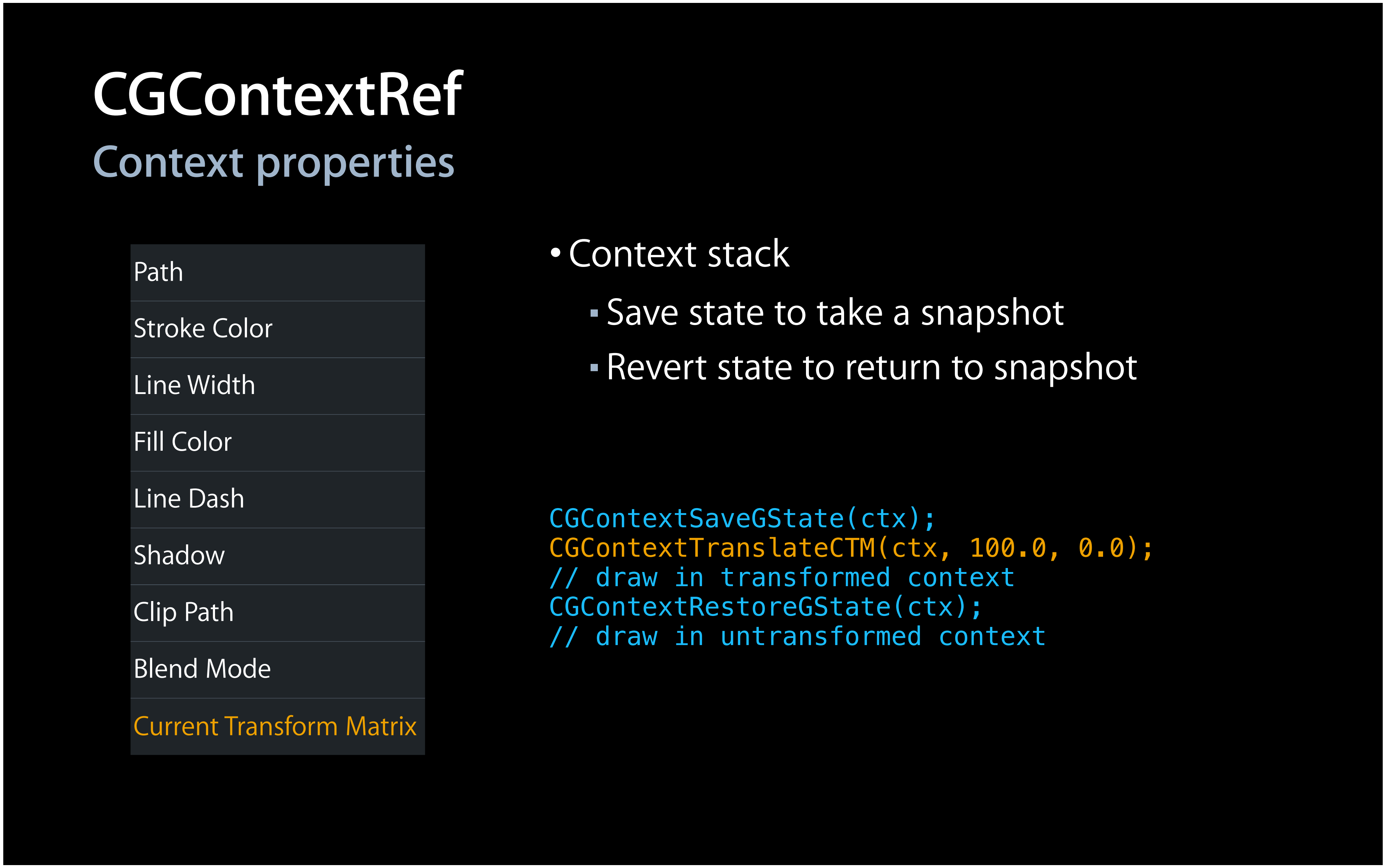
CGContextRef

Context properties

- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix

- Context stack
  - Save state to take a snapshot
  - Revert state to return to snapshot

```
CGContextSaveGState(ctx);
CGContextTranslateCTM(ctx, 100.0, 0.0);
// draw in transformed context
CGContextRestoreGState(ctx);
// draw in untransformed context
```



CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix

- Context stack
  - Save state to take a snapshot
  - Revert state to return to snapshot

```
CGContextSaveGState(ctx);
CGContextTranslateCTM(ctx, 100.0, 0.0);
// draw in transformed context
CGContextRestoreGState(ctx);
// draw in untransformed context
```

CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix

- Context stack
  - Save state to take a snapshot
  - Revert state to return to snapshot

```
CGContextSaveGState(ctx);
CGContextTranslateCTM(ctx, 100.0, 0.0);
// draw in transformed context
CGContextRestoreGState(ctx);
// draw in untransformed context
```

CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix

- Context stack
  - Save state to take a snapshot
  - Revert state to return to snapshot

```
CGContextSaveGState(ctx);
CGContextTranslateCTM(ctx, 100.0, 0.0);
// draw in transformed context
CGContextRestoreGState(ctx);
// draw in untransformed context
```

Don’t Call Us, We’ll Call You

No context, no drawing!

```
- (void)tapGestureReceived:(UIT     stureRe     izer *)tapGR {
    // process event...
    CGContextRef ctx = UIGraphicsC      r     ontext();

    CGGradientRef gradient = [self g      nt];

    CGPoint startPoint = CGPointMa      G     GetMidX(self.bounds), 0.0);
    CGPoint endPoint = CGPointMa     Rec     MidX(self.bounds),
                                    CGRect     xY(self.bounds));

    CGContextDrawLinearGradien     x, gradie     startPoint, endPoint, 0);
}
```

Don’t Call Us, We’ll Call You

State changed, time to redraw

```
- (void)tapGestureReceived:(UITapGestureR    gnizer *)tapGR {
    // process event...
    [[tapGR view] setNeedsDisplay];
}
```

Don’t Call Us, We’ll Call You

UIKit provides the context

```
- (void)drawRect:(CGRect)rect {
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGGradientRef gradient = [self gradient];

    CGPoint startPoint = CGPointMake(CGRectGetMidX(self.bounds), 0.0);
    CGPoint endPoint = CGPointMake(CGRectGetMidX(self.bounds),
                                   CGRectGetMaxY(self.bounds));
    CGContextDrawLinearGradient(ctx, gradient, startPoint, endPoint, 0);
}
```

Well, You Can Call

You create a context

```
- (UIImage *)scaleImage:(UIImage *)image toSize:(CGSize)newSize {
    UIGraphicsBeginImageContextWithOptions(newSize, YES, 0.0);
    CGRect imageRect = {{0.0, 0.0}, newSize};
    [image drawInRect:imageRect];
    UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return scaledImage;
}
```

Well, You Can Call

You create a context

```
- (UIImage *)scaleImage:(UIImage *)image toSize:(CGSize)newSize {
    UIGraphicsBeginImageContextWithOptions(newSize, YES, 0.0);
    CGRect imageRect = {{0.0, 0.0}, newSize};
    [image drawInRect:imageRect];
    UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return scaledImage;
}
```

Well, You Can Call

You create a context

```
- (UIImage *)scaleImage:(UIImage *)image toSize:(CGSize)newSize {
    UIGraphicsBeginImageContextWithOptions(newSize, YES, 0.0);
    CGRect imageRect = {{0.0, 0.0}, newSize};
    [image drawInRect:imageRect];
    UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return scaledImage;
}
```

Well, You Can Call

You create a context

```
- (UIImage *)scaleImage:(UIImage *)image toSize:(CGSize)newSize {
    UIGraphicsBeginImageContextWithOptions(newSize, YES, 0.0);
    CGRect imageRect = {{0.0, 0.0}, newSize};
    [image drawInRect:imageRect];
    UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return scaledImage;
}
```

Creating a BitMap Context

`CGBitmapContextCreate`

- Size
- Scale
- Color space
- Bytes per row
- Bits per component
- Alpha channel
- Alpha-component location
- Byte order (big or little endian)

`UIGraphicsBeginImageContextWithOptions`

- Size
- Opacity
- Scale

Creating a BitMap Context

`UIGraphicsBeginImageContextWithOptions`

- Use unless you have a custom algorithm

`CGBitmapContextCreate`

- Use when your custom algorithm requires

Basics

Wrap-up

- Context is geometric space in which to draw
- Translates from geometric description to pixels
- Current transform matrix defines the space
- Don’t call us—we will call you when you tell us to

Paths

Path Primitives

Points, lines, arcs, and curves

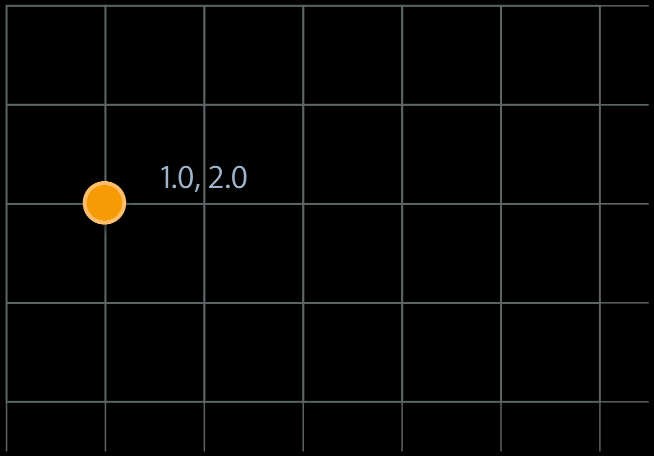
Path Primitives

Points, lines, arcs, and curves

```
UIBezierPath *path = [UIBezierPath bezierPath];
CGPoint startPoint = CGPointMake(1.0, 2.0);
[path moveToPoint:startPoint];
CGPoint nextPoint = CGPointMake(4.0, 2.0);
[path addLineToPoint:nextPoint];
[path setLineWidth:1.0];
[path stroke];
```

Path Primitives

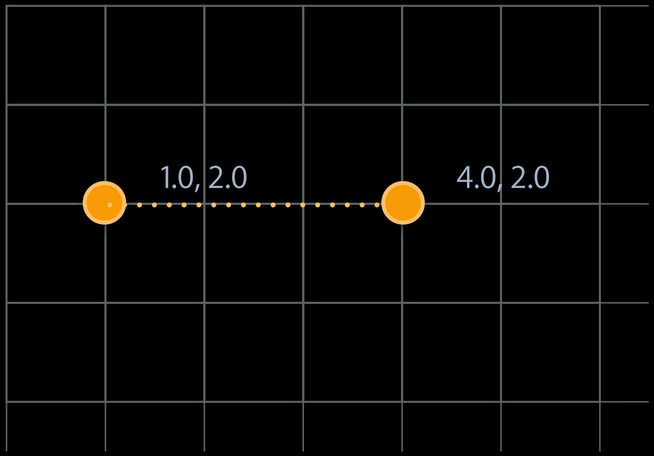
Points, lines, arcs, and curves



```
UIBezierPath *path = [UIBezierPath bezierPath];
CGPoint startPoint = CGPointMake(1.0, 2.0);
[path moveToPoint:startPoint];
CGPoint nextPoint = CGPointMake(4.0, 2.0);
[path addLineToPoint:nextPoint];
[path setLineWidth:1.0];
[path stroke];
```

Path Primitives

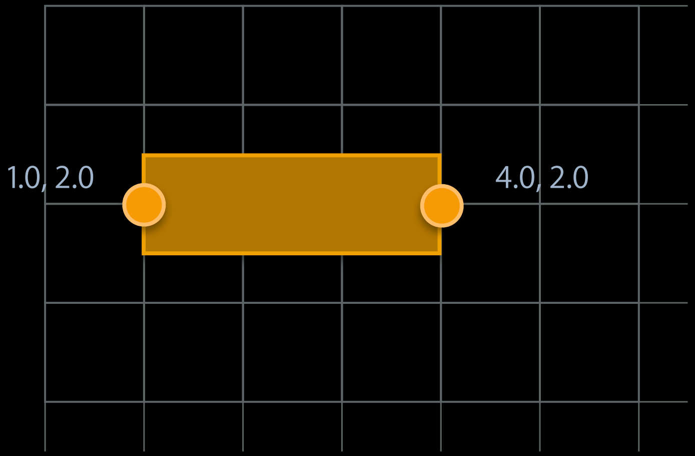
Points, lines, arcs, and curves



```
UIBezierPath *path = [UIBezierPath bezierPath];
CGPoint startPoint = CGPointMake(1.0, 2.0);
[path moveToPoint:startPoint];
CGPoint nextPoint = CGPointMake(4.0, 2.0);
[path addLineToPoint:nextPoint];
[path setLineWidth:1.0];
[path stroke];
```

Path Primitives

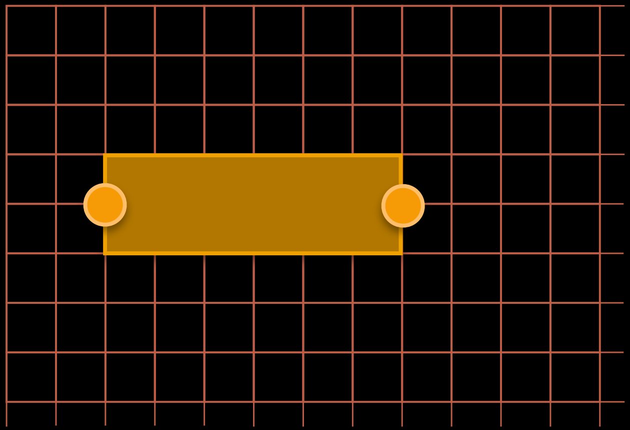
Points, lines, arcs, and curves



```
UIBezierPath *path = [UIBezierPath bezierPath];
CGPoint startPoint = CGPointMake(1.0, 2.0);
[path moveToPoint:startPoint];
CGPoint nextPoint = CGPointMake(4.0, 2.0);
[path addLineToPoint:nextPoint];
[path setLineWidth:1.0];
[path stroke];
```

Anti-aliasing

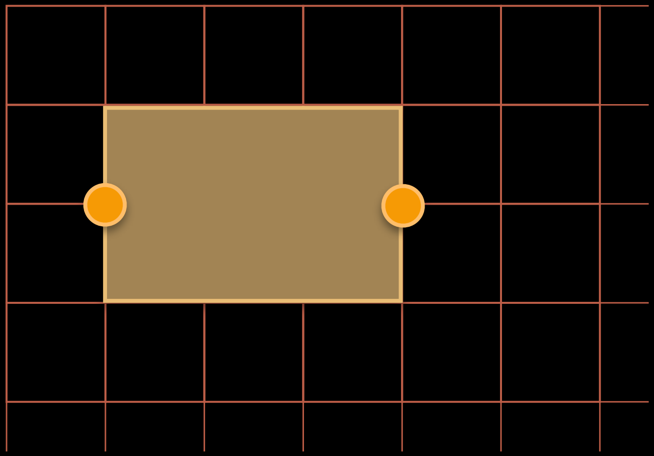
When pixels become important



- Retina Display will be full intensity
- Two pixels wide

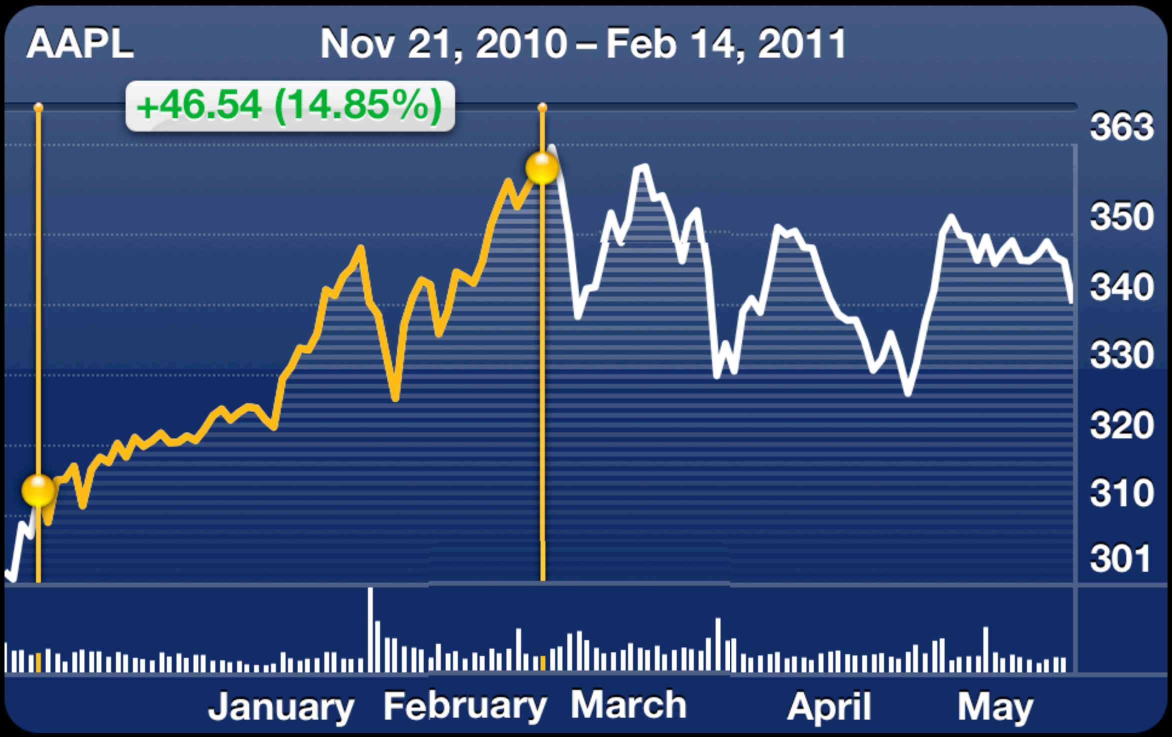
Anti-aliasing

When pixels become important

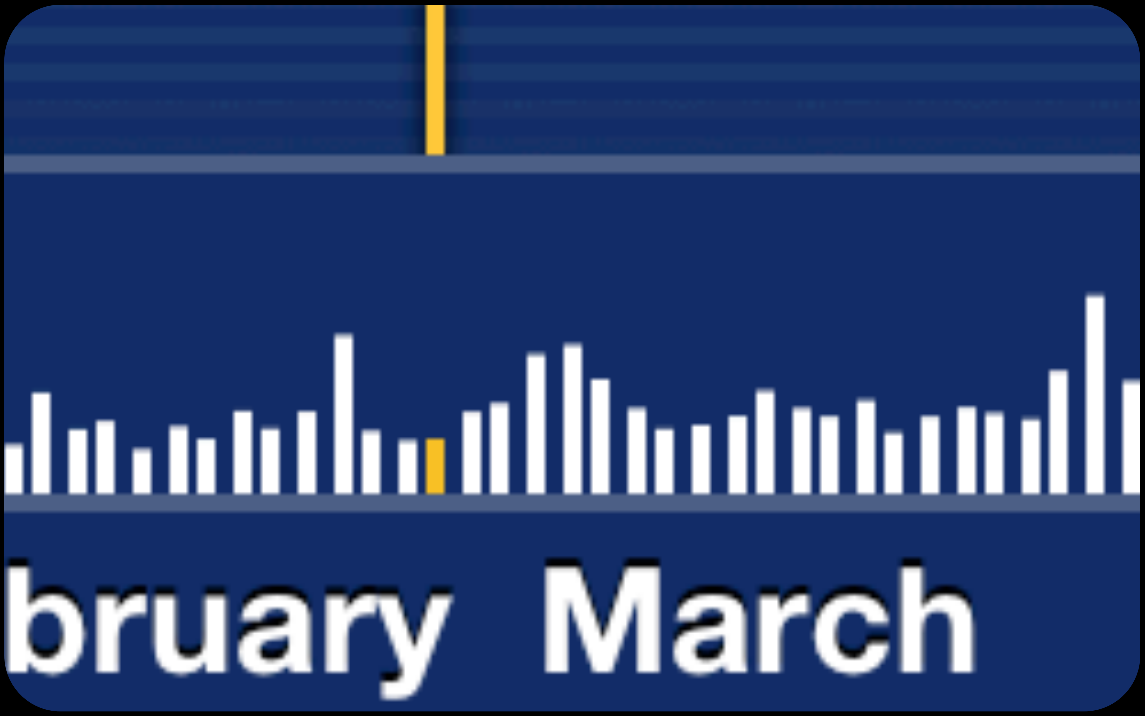


- Half the intensity
- Twice the coverage

Pixel-Perfect Drawing

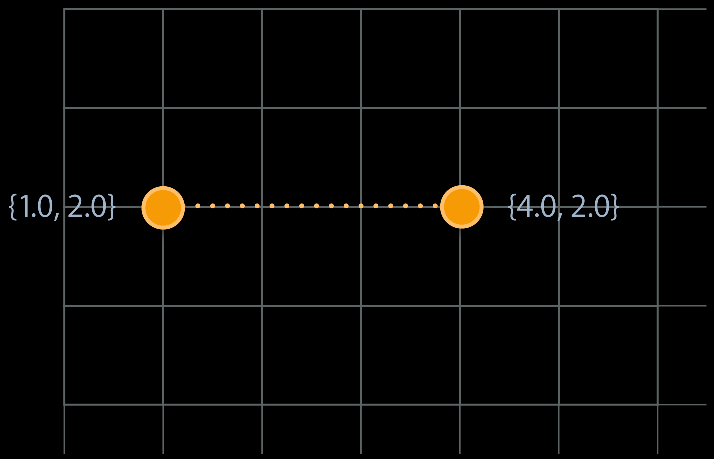


Pixel-Perfect Drawing



Pixel-Perfect Drawing

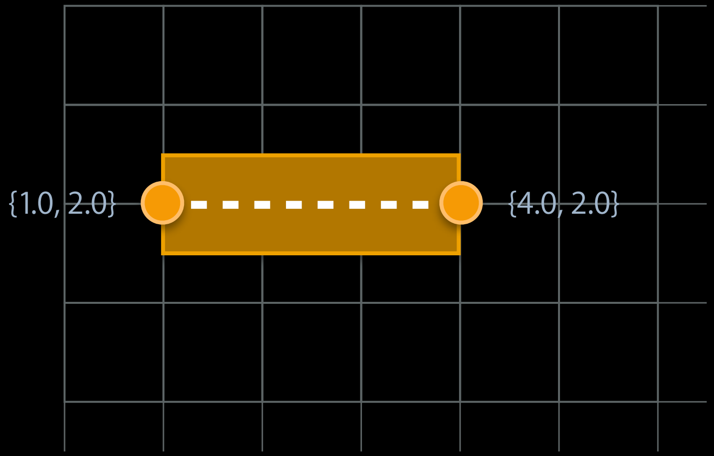
Sometimes you need to think about pixels



- Line width—1 point
- Move to—{1.0, 2.0}
- Add line to—{4.0, 2.0}

Pixel-Perfect Drawing

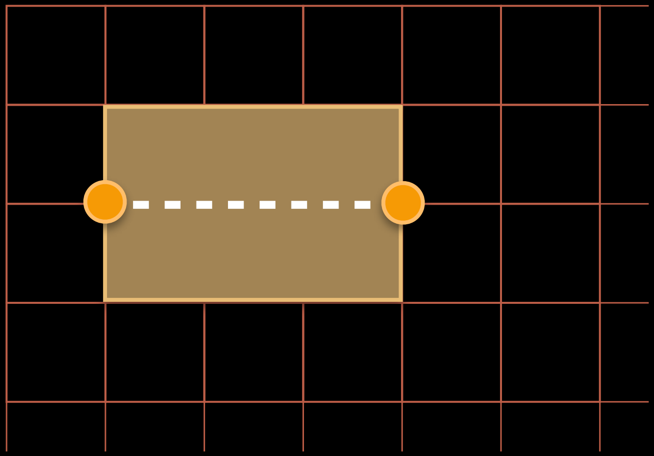
Sometimes you need to think about pixels



- What you want from your geometric thinking

Pixel-Perfect Drawing

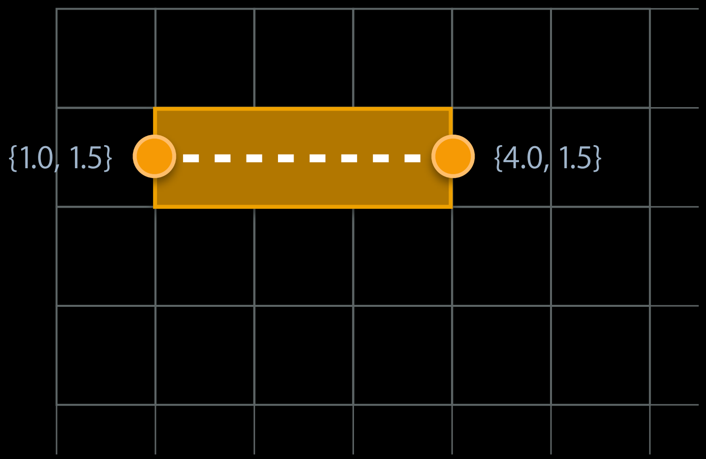
Sometimes you need to think about pixels



- Paths paint half the width on each side of the line
- What you get
- Anti-aliased
    - Twice the width/half the intensity

Pixel-Perfect Drawing

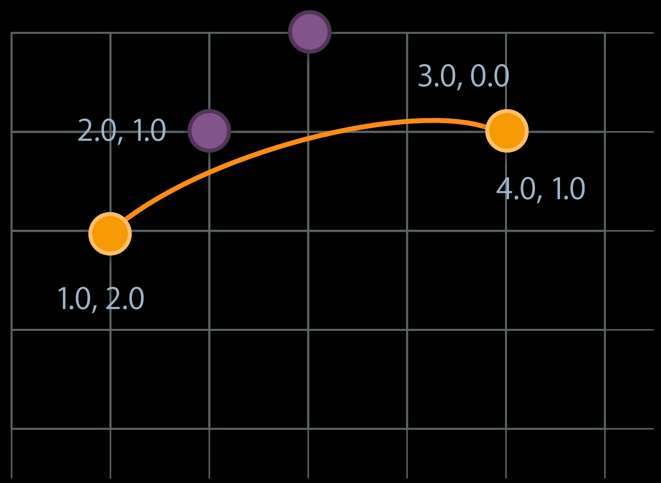
Sometimes you need to think about pixels



- Odd line widths

Pixel-Perfect Drawing

Sometimes you can't



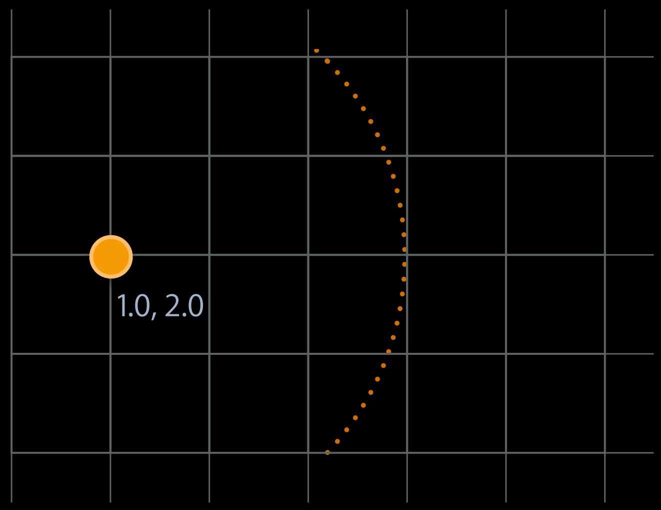
- Curves do not line up exactly with pixels, so don't try, just let Quartz do its magic

Think Geometrically!

Unless you need to be pixel perfect

Path Primitives

Points, lines, arcs, and curves

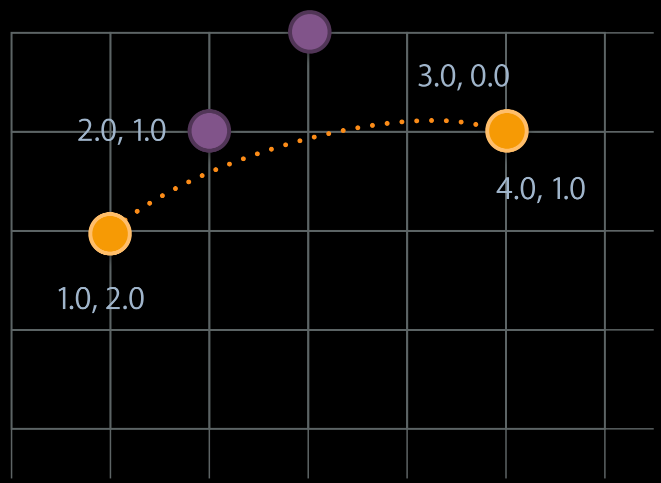


- Arc of radius
  - Centered at a point
  - Extended through some angle

```
UIBezierPath *path = [UIBezierPath bezierPath];
CGPoint center = CGPointMake(1.0, 2.0);
[path addArcWithCenter:center radius:3.0 startAngle:-0.25 * M_PI
              endAngle:0.25 * M_PI clockwise:YES];
```

Path Primitives

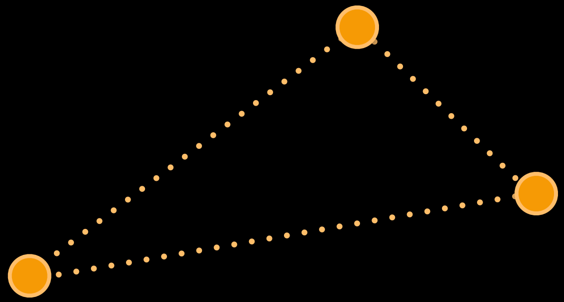
Points, lines, arcs, and curves



- Curves
  - Two points
  - Two control points

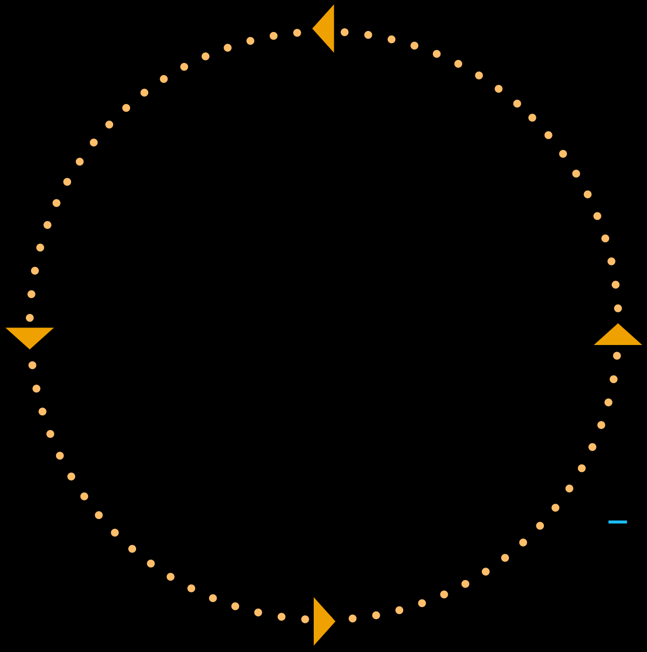
```
UIBezierPath *path = [UIBezierPath bezierPath];
// declare and init start, end, control1 and control2
[path moveToPoint:start];
[path addCurveToPoint:end controlPoint1:control1 controlPoint2:control2];
[path stroke];
```

Closing a Path



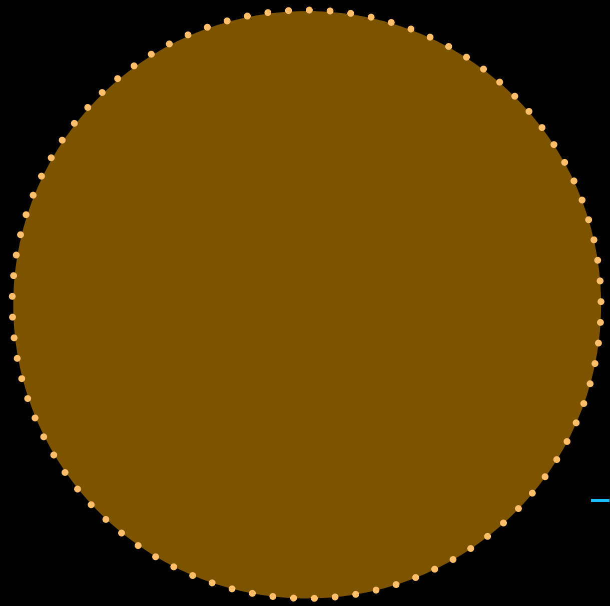
```
- (UIBezierPath *)triangleWithPoints:(CGPoint *)points {
  UIBezierPath *path = [UIBezierPath bezierPath];
  [path moveToPoint:points[0]];
  [path addLineToPoint:points[1]];
  [path addLineToPoint:points[2]];
  [path closePath];
  return path;
}
```

Filling a Path



```
- (void)fillCircleCenteredAt:(CGPoint)center {
  UIBezierPath *path = [UIBezierPath bezierPath];
  [path addArcWithCenter:center radius:50.0
              startAngle:0.0 endAngle:2.0 * M_PI
               clockwise:NO];
  [path fill];
}
```

Filling a Path



```
- (void)fillCircleCenteredAt:(CGPoint)center {
  UIBezierPath *path = [UIBezierPath bezierPath];
  [path addArcWithCenter:center radius:50.0
              startAngle:0.0 endAngle:2.0 * M_PI
               clockwise:NO];
  [path fill];
}
```

Clipping with a Path

Rounded rectangles without the mask

Clipping with a Path

Rounded rectangles without the mask

```
CGRect rect = CGRectMake(25.0, 25.0, 300.0, 225.0);
UIBezierPath  *path = [UIBezierPath bezierPathWithRoundedRect:rect
                                    byRoundingCorners:UIRectCornerAllCorners
                                    cornerRadii:radii];
[path addClip];
[[self image] drawAtPoint:point];
```

Clipping with a Path

Rounded rectangles without the mask

```
CGRect rect = CGRectMake(25.0, 25.0, 300.0, 225.0);
UIBezierPath  *path = [UIBezierPath bezierPathWithRoundedRect:rect
                                    byRoundingCorners:UIRectCornerAllCorners
                                    cornerRadii:radii];
[path addClip];
[[self image] drawAtPoint:point];
```

Clipping with a Path

Rounded rectangles without the mask

```
CGRect rect = CGRectMake(25.0, 25.0, 300.0, 225.0);
UIBezierPath  *path = [UIBezierPath bezierPathWithRoundedRect:rect
                                    byRoundingCorners:UIRectCornerAllCorners
                                    cornerRadii:radii];
[path addClip];
[[self image] drawAtPoint:point];
```

Clipping with a Path

Rounded rectangles without the mask

```
CGRect rect = CGRectMake(25.0, 25.0, 300.0, 225.0);
UIBezierPath  *path = [UIBezierPath bezierPathWithRoundedRect:rect
                                    byRoundingCorners:UIRectCornerAllCorners
                                    cornerRadii:radii];
[path addClip];
[[self image] drawAtPoint:point];
```

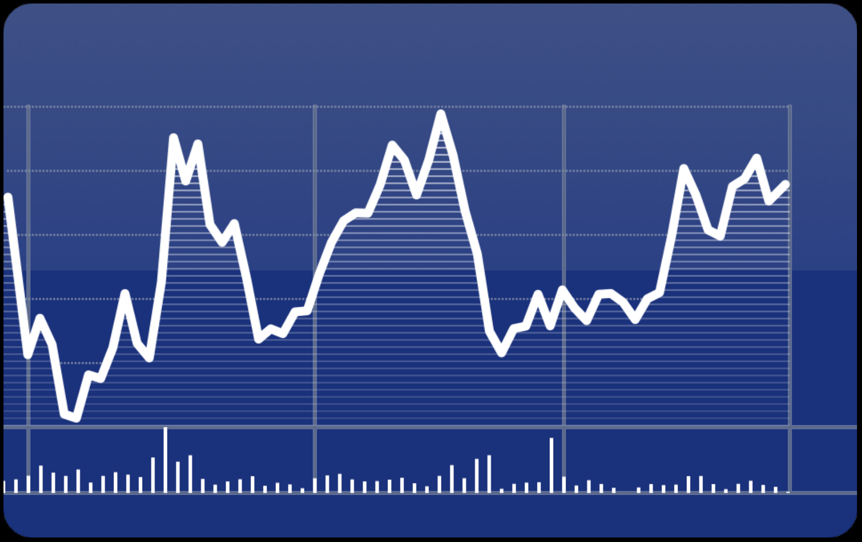
Clipping with a Path

Rounded rectangles without the mask

Demo

Contexts, paths, points, and pixels

Demo Wrap-Up



- Linear gradient
- Clipping
- Crisp lines

Text Drawing

Drawing Text

Two options

- NSString UIKit drawing additions
  - Simple
  - Geared toward UI element text
- Core Text
  - Full-featured text layout and drawing
  - Aimed at complex text-display features

Layout Information

Font-based text metrics

- Single line

```
CGSize ligerSize = [@”It’s a Liger” sizeWithFont:font];
```

Layout Information

Font-based text metrics

- Multiline

```
CGSize textSize = [warAndPeace sizeWithFont:[UIFont systemFontOfSize:14.0]
                          constrainedToSize:CGSizeMake(124.0, 256.0)];
```

Drawing Text

Single line

```
CGSize ligerSize = [@”It’s a Liger” drawAtPoint:CGPointMake(42.0, 127.0)
                                          withFont:font];
```

Drawing Text

Multiline

```
CGSize warAndPeaceSize = [warAndPeace drawInRect:textRect
                                     withFont:font];
```

Drawing Text

No context, no drawing

```
- (void)tapGestureReceived:(UITapG         ognizer *)tapGR {
    // process event...
    [@”Lower Rack” drawAtPoint:CGPoi       e(100.0, 100.0)
                       withFont:font
}
```

Drawing Text

State changed, time to redraw

```
- (void)tapGestureReceived:(UITapGestur    cognizer *)tapGR {
    // process event...
    [[tapGR view] setNeedsDisplay
}
```

Core Text

Full-featured text layout

- Use Core Text for anything more than a couple of lines of text

Advanced Text Handling for iPhone OS	ADC on iTunes U

Shadows

CGContextRef

Context properties

Path
Stroke Color
Line Width
Fill Color
Line Dash
Shadow
Clip Path
Blend Mode
Current Transform Matrix

- Shadow properties
  - Offset
  - Color
  - Blur radius

```
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
                  CGSizeMake(1.0, -shadowHeight), 0.0,
                  [[UIColor darkGrayColor] CGColor]);
// drawing gets shadowed
```

CGContextRef

Context properties

- Path
- Stroke Color
- Line Width
- Fill Color
- Line Dash
- Shadow
- Clip Path
- Blend Mode
- Current Transform Matrix

- Shadow properties
  - Offset
  - Color
  - Blur radius

```
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
               CGSizeMake(1.0, -shadowHeight), 0.0,
               [[UIColor darkGrayColor] CGColor]);
// drawing gets shadowed
CGContextRestoreGState(ctx);
// shadow state is reverted
```

Shadowed Text

```
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor darkGrayColor] CGColor]);
[@”March” drawInRect:monthRect withFont:font];
CGContextRestoreGState(ctx);
// shadow state is reverted
```

Shadowed Text

```
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor darkGrayColor] CGColor]);
[@”March” drawInRect:monthRect withFont:font];
CGContextRestoreGState(ctx);
// shadow state is reverted
```

Shadowed Text

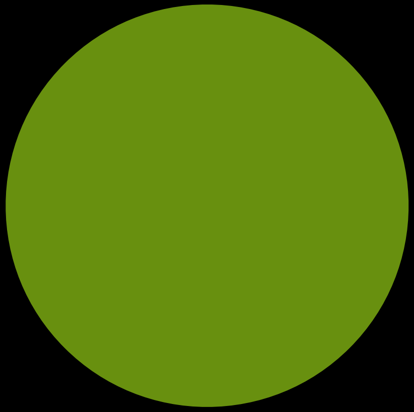
```
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor orangeColor] CGColor]);
[@”March” drawInRect:monthRect withFont:font];
CGContextRestoreGState(ctx);
// shadow state is reverted
```

Shadowed Text

```
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor orangeColor] CGColor]);
[@”March” drawInRect:monthRect withFont:font];
CGContextRestoreGState(ctx);
// shadow state is reverted
```

Painter’s Algorithm

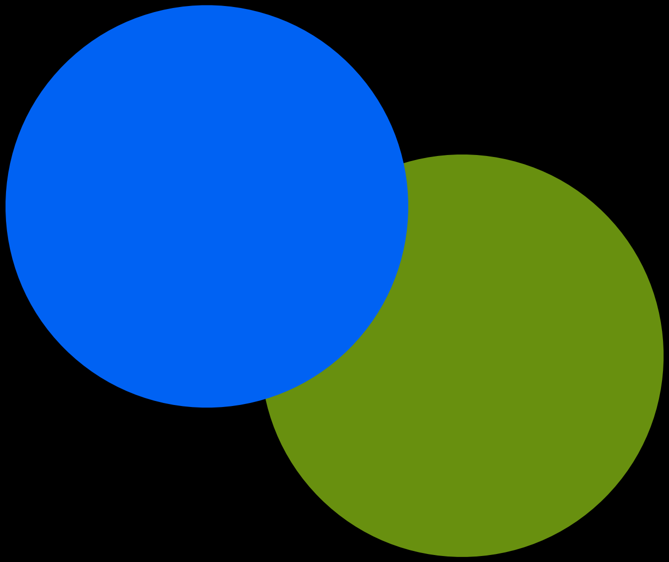
Last color down wins



```
UIColor *color1 = ...
UIColor *color2 = ...
UIColor *color3 = ...
CGRect square = CGRectMake(0.0, 0.0, 100.0,
100.0);
UIBezierPath *circle = [UIBezierPath
           bezierPathWithOvalInRect:square];
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
```

Painter's Algorithm

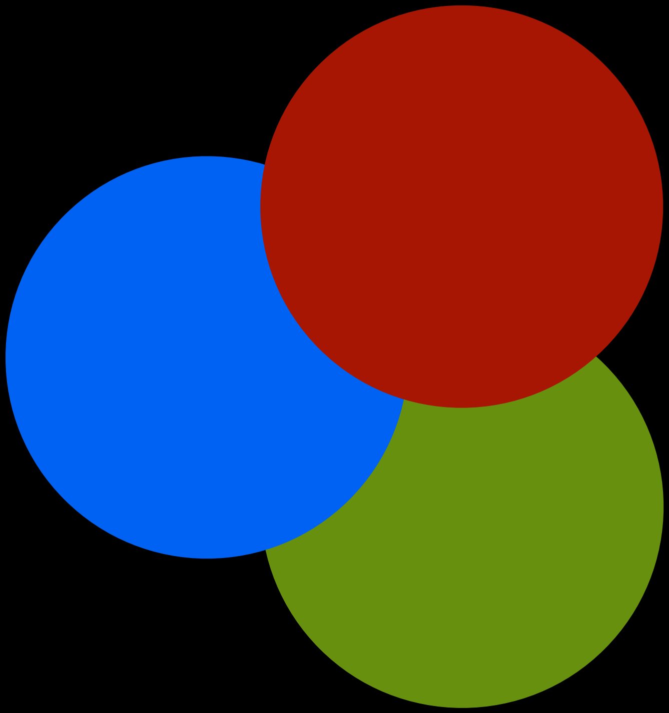
Last color down wins



```
UIColor *color1 = ...
UIColor *color2 = ...
UIColor *color3 = ...
CGRect square = CGRectMake(0.0, 0.0, 100.0,
100.0);
UIBezierPath *circle = [UIBezierPath
           bezierPathWithOvalInRect:square];
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
```

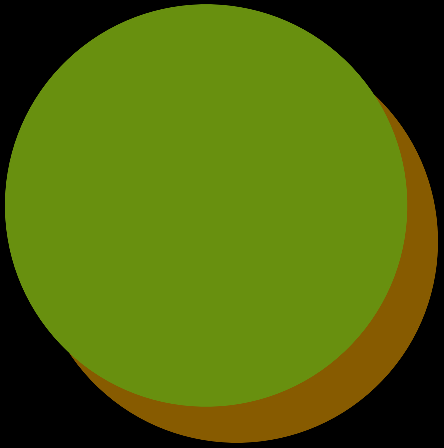
Painter’s Algorithm

Last color down wins



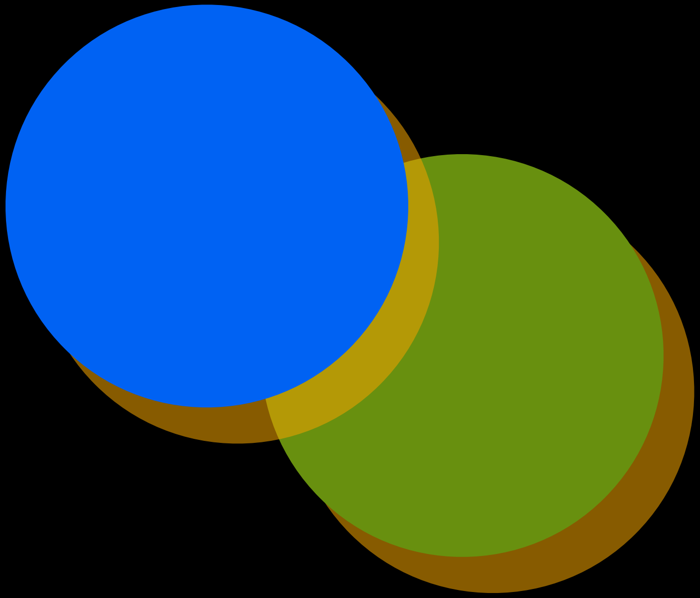
```
UIColor *color1 = ...
UIColor *color2 = ...
UIColor *color3 = ...
CGRect square = CGRectMake(0.0, 0.0, 100.0,
100.0);
UIBezierPath *circle = [UIBezierPath
          bezierPathWithOvalInRect:square];
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
```

```
...
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor orangeColor] CGColor]);
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
CGContextRestoreGState(ctx);
```



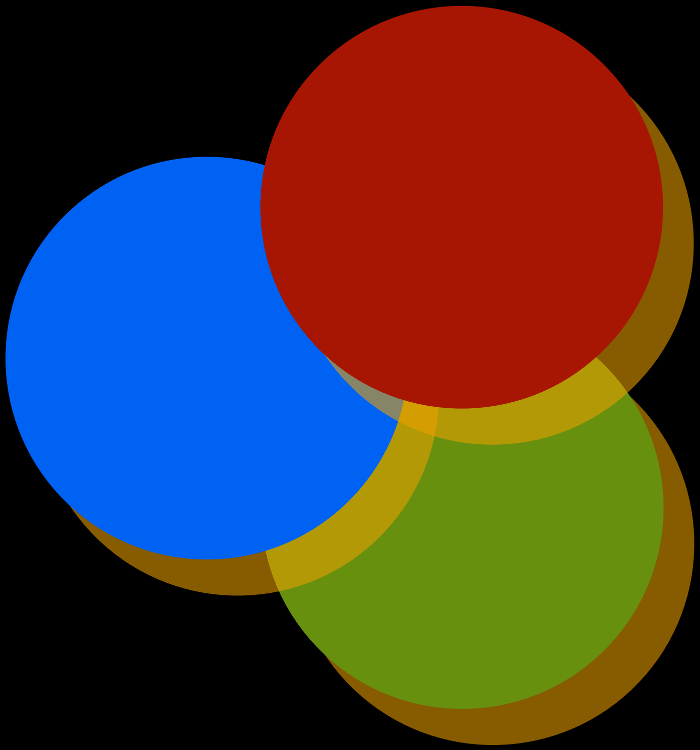
```
...
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor orangeColor] CGColor]);
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
CGContextRestoreGState(ctx);
```

Multiple Draws Result in Multiple Shadows



```
...
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor orangeColor] CGColor]);
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
CGContextRestoreGState(ctx);
```

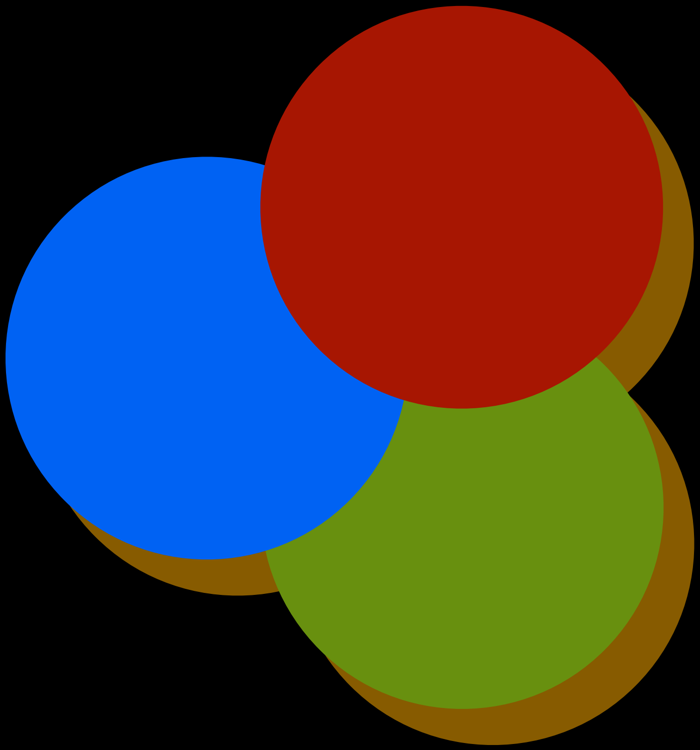
Multiple Draws Result in Multiple Shadows



```
...
CGContextSaveGState(ctx);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
          CGSizeMake(1.0, -shadowHeight), 0.0,
          [[UIColor orangeColor] CGColor]);
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
CGContextRestoreGState(ctx);
```

Transparency Layers

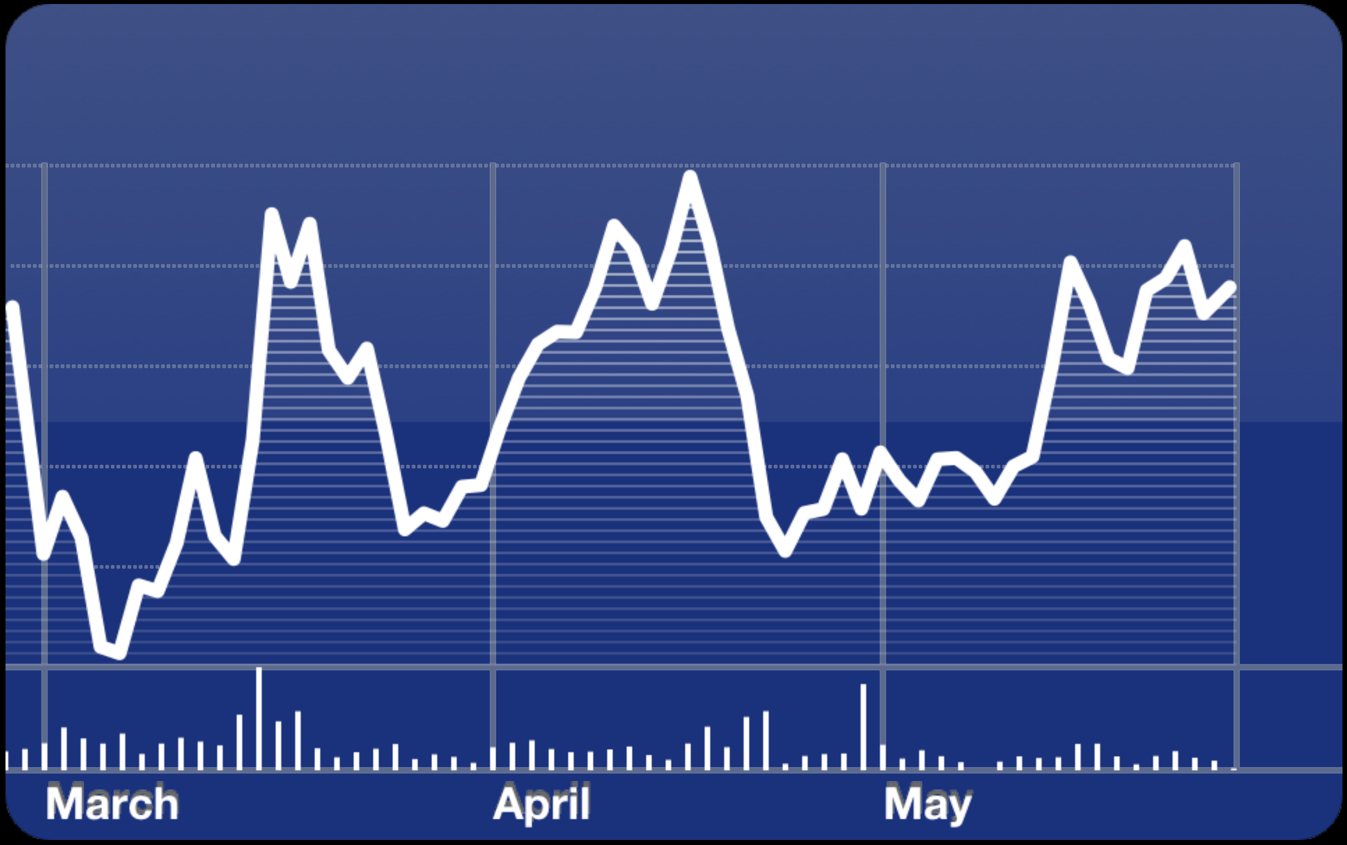
One group—one shadow



```
...
CGContextSaveGState(ctx);
CGContextBeginTransparencyLayer(ctx, NULL);
CGFloat shadowHeight = 2.0;
CGContextSetShadowWithColor(ctx,
         CGSizeMake(1.0, -shadowHeight), 0.0,
         [[UIColor orangeColor] CGColor]);
[color1 setFill];
[circle fill];
[color2 setFill];
[circle fill];
[color3 setFill];
[circle fill];
CGContextEndTransparencyLayer(ctx);
CGContextRestoreGState(ctx);
```

Demo

Text and shadows


March
April
May

Images

UIImage

- Use UIImageView
- If you must, you can draw an image with:

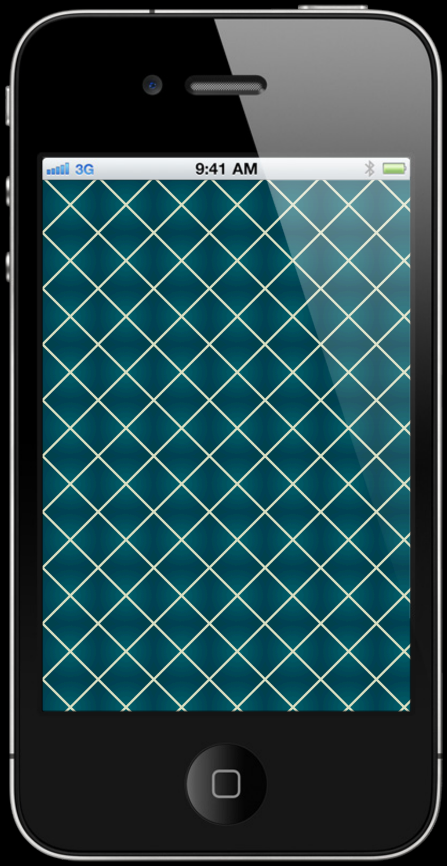
```
[[self myImage] drawAtPoint:location];

[[self myImage] drawInRect:location];
```

Patterns

Patterns

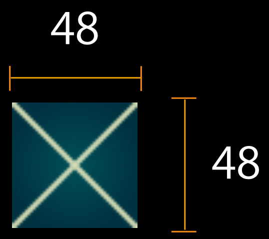
Repeating an image



- Single image
- Drawn repeatedly

Patterns

Repeating an image



- Single pattern tile
  - Radial gradient background
  - Lines drawn across corners
- Drawn with Quartz
  - Normally, an artist would create

Creating a Pattern

UIKit to the rescue again

```
@implementation MyViewController
...
- (void)viewDidLoad {
    [super viewDidLoad];
    CGSize patternSize = CGSizeMake(48.0, 48.0);
    UIImage *patternImage = [self patternImageOfSize:patternSize];
    self.view.backgroundColor = [UIColor colorWithPatternImage:patternImage];
}
...
@end
```

Using a Pattern

UIKit to the rescue again

- Setting `backgroundColor` instead of using `drawRect:` is more efficient
- But don't be afraid of `drawRect:`

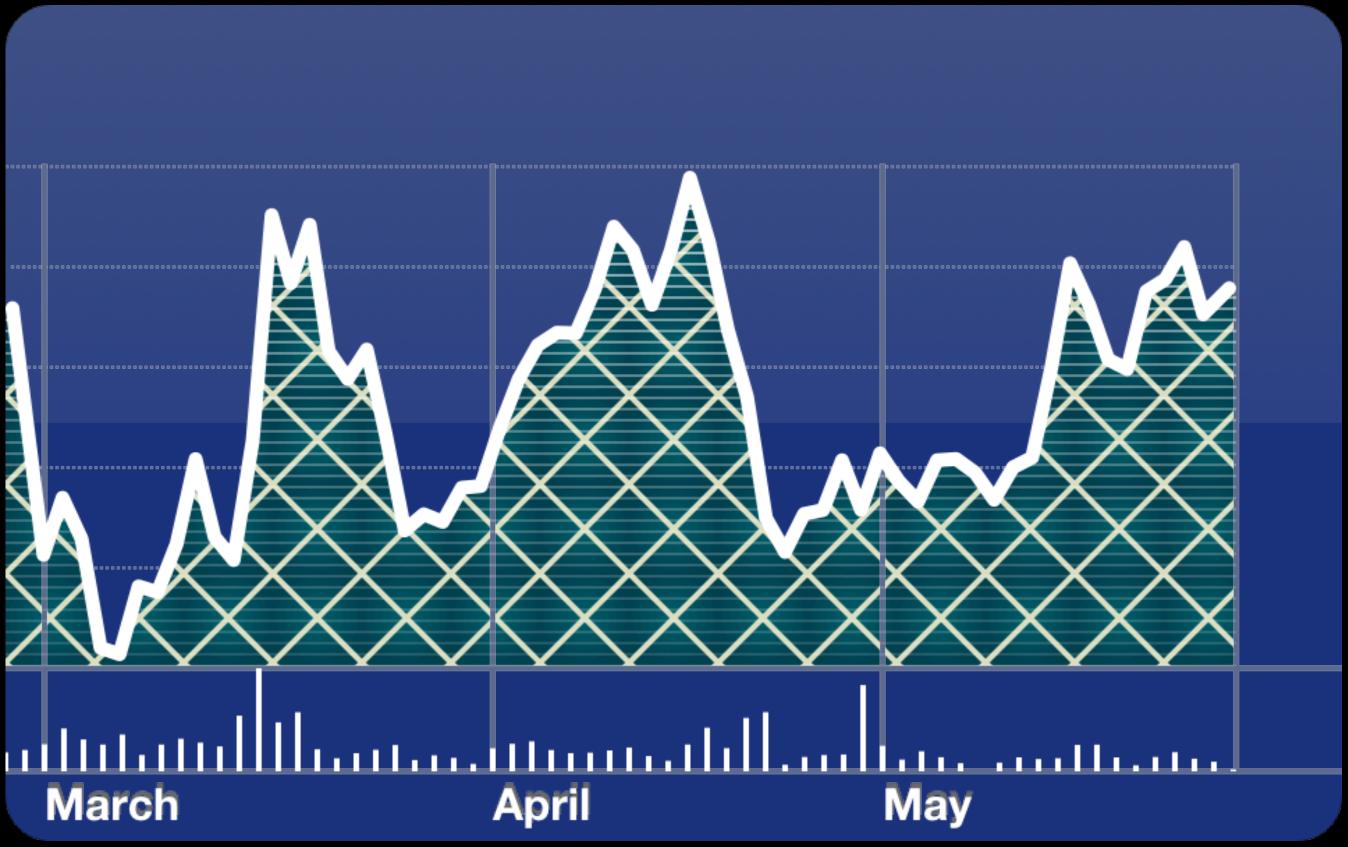
```
@implementation MyViewController
...
- (void)viewDidLoad {
    [super viewDidLoad];
    CGSize patternSize = CGSizeMake(48.0, 48.0);
    UIImage *patternImage = [self patternImageOfSize:patternSize];
    self.view.backgroundColor = [UIColor colorWithPatternImage:patternImage];
}
...
@end
```

Homework

Stroke a path with the 48x48 pattern

Demo

Patterns


March
April
May

Summary

- You can draw that with Quartz!
- Draw in a context, no active context, no drawing
- Think geometrically

More Information

Bill Dudney
Frameworks Evangelist
dudney@apple.com

Documentation
Quartz 2D Programming Guide
http://developer.apple.com/library/ios/#documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/Introduction/Introduction.html

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

Related Sessions

Understanding UIKit Rendering	Mission Thursday 10:15AM
Advanced Text Handling for iPhone OS	ADC on iTunes U

Labs

Drawing on iOS Lab	Application Frameworks Lab D Friday 9:00AM

WWDC2011