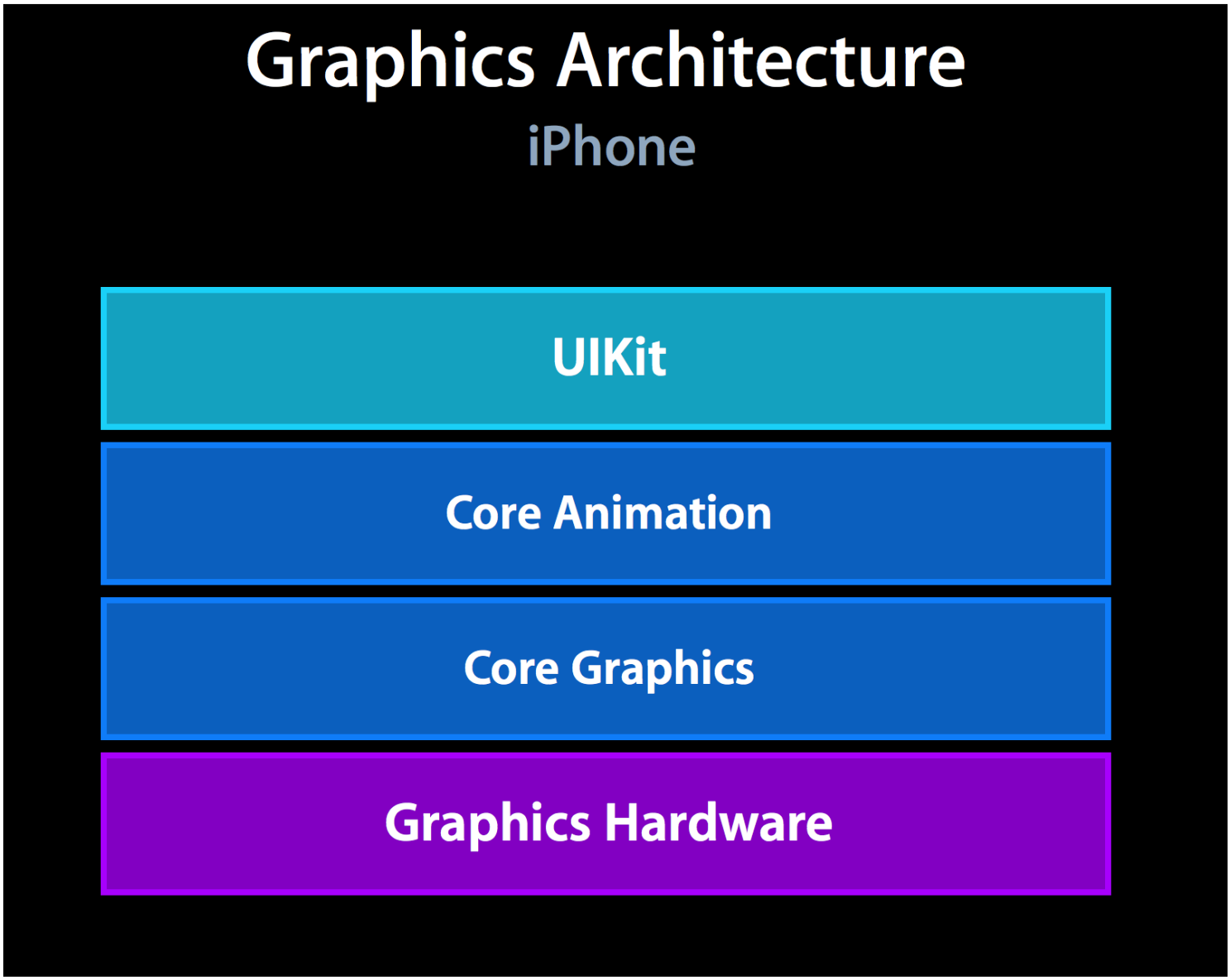
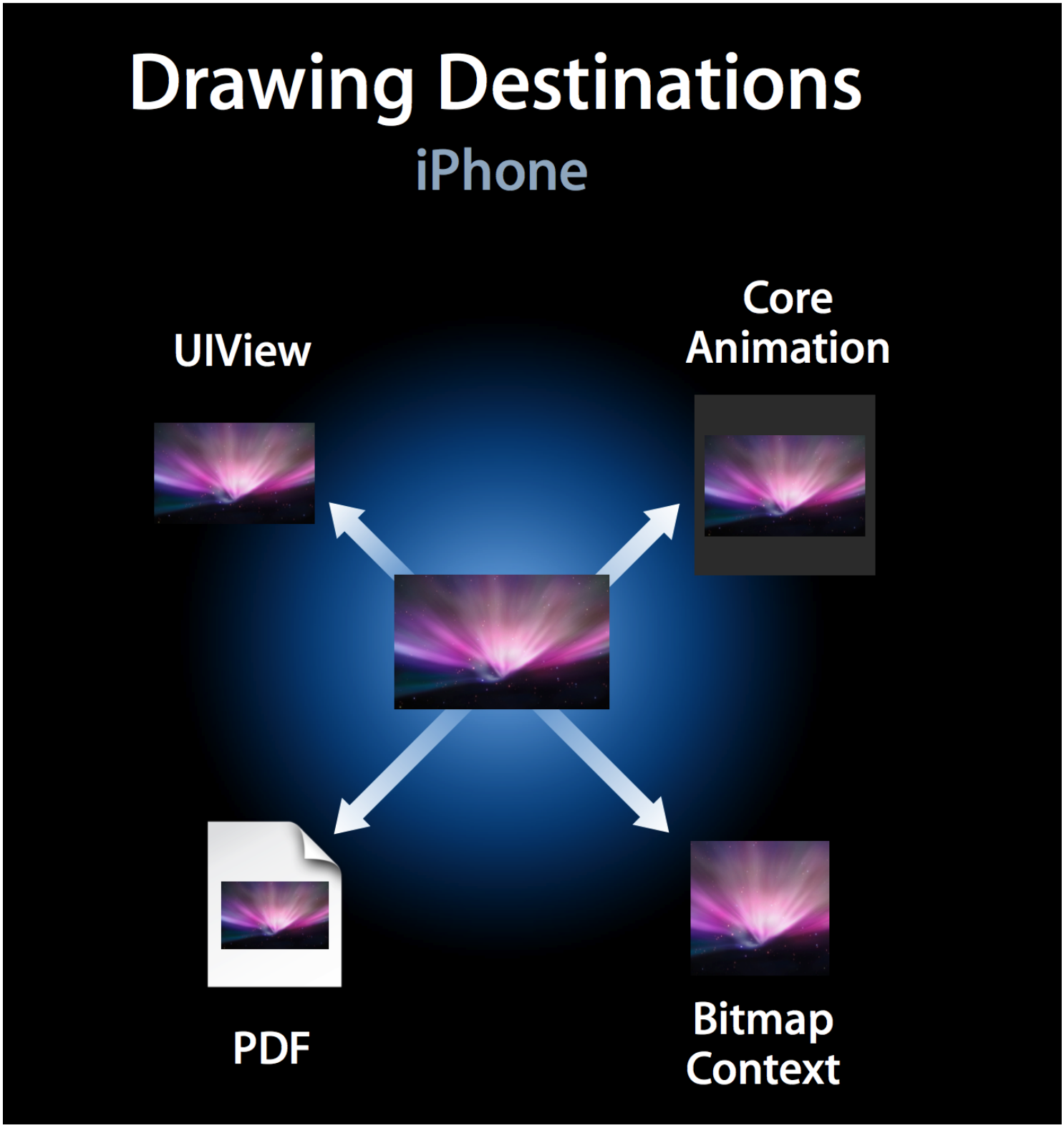
Core Graphics - 101



Core Graphics Key Features

- Modern 2D Graphics API
- Device independence
- Resolution independence
- High-performance
- Anti-aliased graphics and text
- Wide array of supported image formats
- Built-in color management
- Built-in PDF support

Use Core Graphics do Any One of These Things



CoreGraphics and Quartz 2D

- UIKit offers very basic drawing functionality

```
UIRectFill(CGRect rect);
UIRectFrame(CGRect rect);
```

- CoreGraphics: Drawing APIs
- CG is a C-based API, not Objective-C
- CG and Quartz 2D drawing engine define simple but powerful graphics primitives
  - Graphics context
  - Transformations
  - Paths
  - Colors
  - Fonts
  - Painting operations

Object-Like

Create, access, modify, use, reuse

Images	CGImageCreate CGImageGetWidth, CGImageGetHeight CGContextDrawImage
Colors	CGColorCreate CGContextSetFillColorWithColor
Paths	CGPathCreateMutable CGPathAddRect, CGPathAddArc

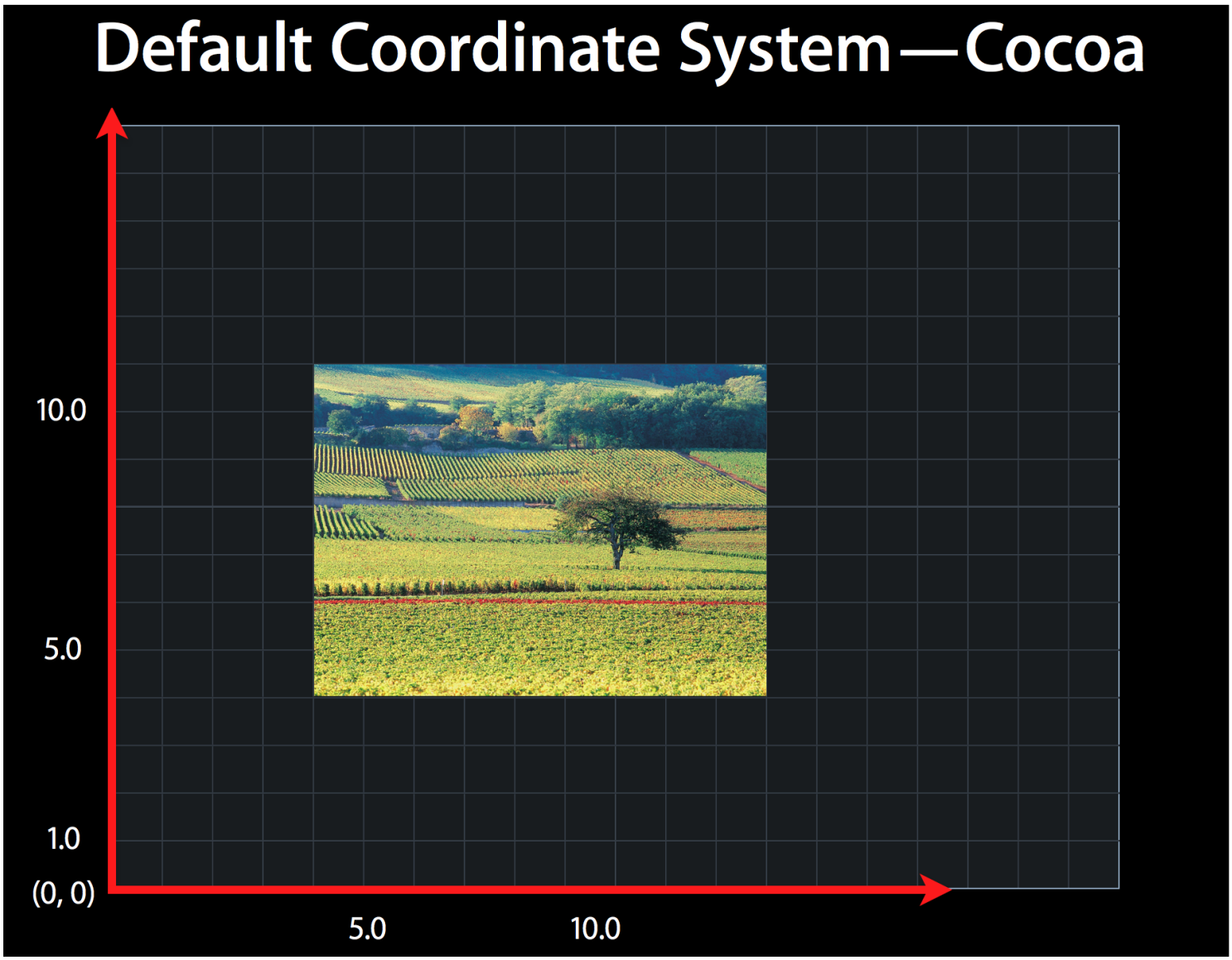
Memory Management

Object ownership

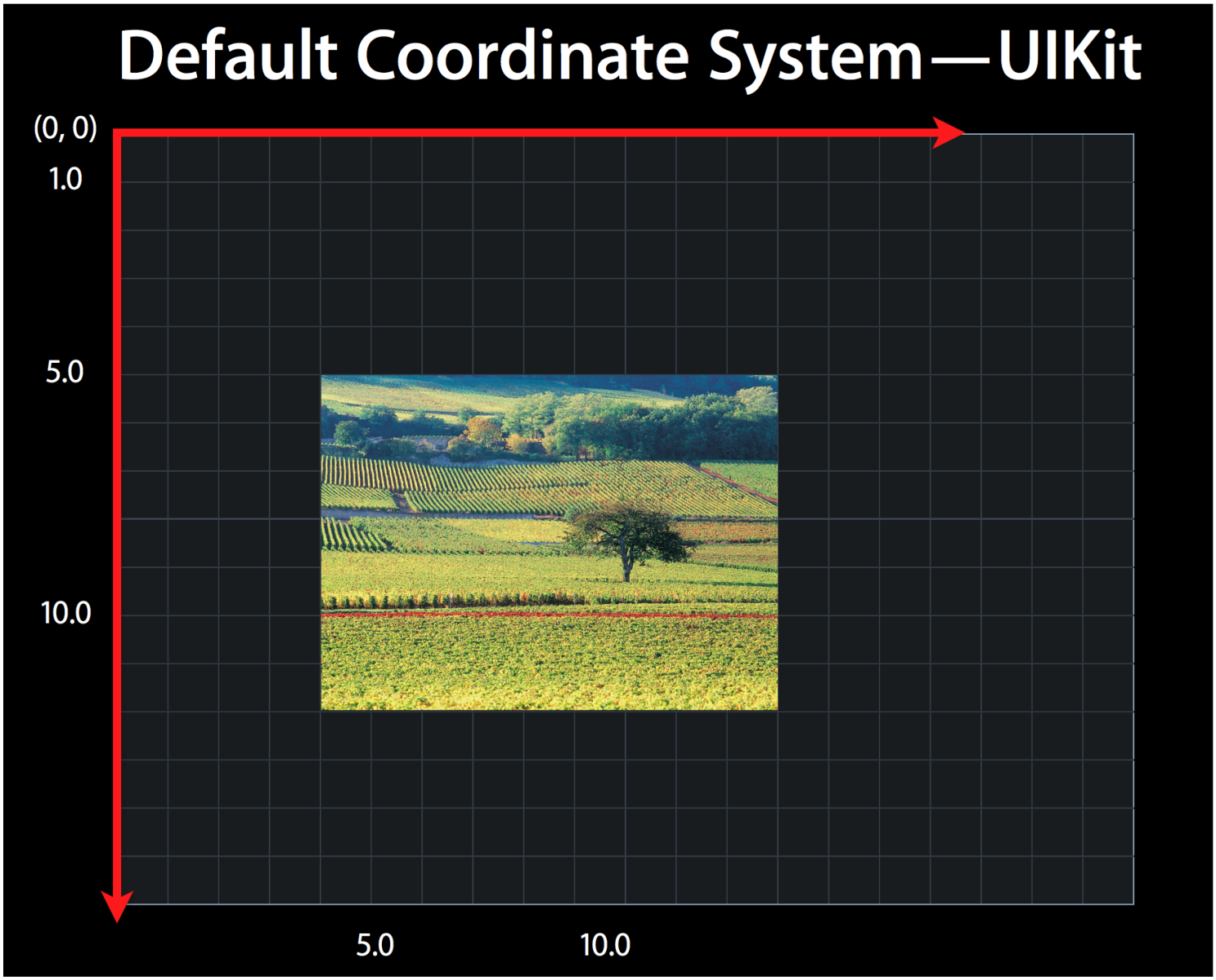
- If you create it, you own it
- Owners must release objects

  `CGImageRelease, CFRelease`
- If you don't own it, do not release it
- Want to keep it? Retain it

  `CGImageRetain, CFRetain`


Default Coordinate System—Cocoa
10.0
5.0
1.0
(0, 0)
5.0
10.0

UIKit Coordinates



Coordinate Space

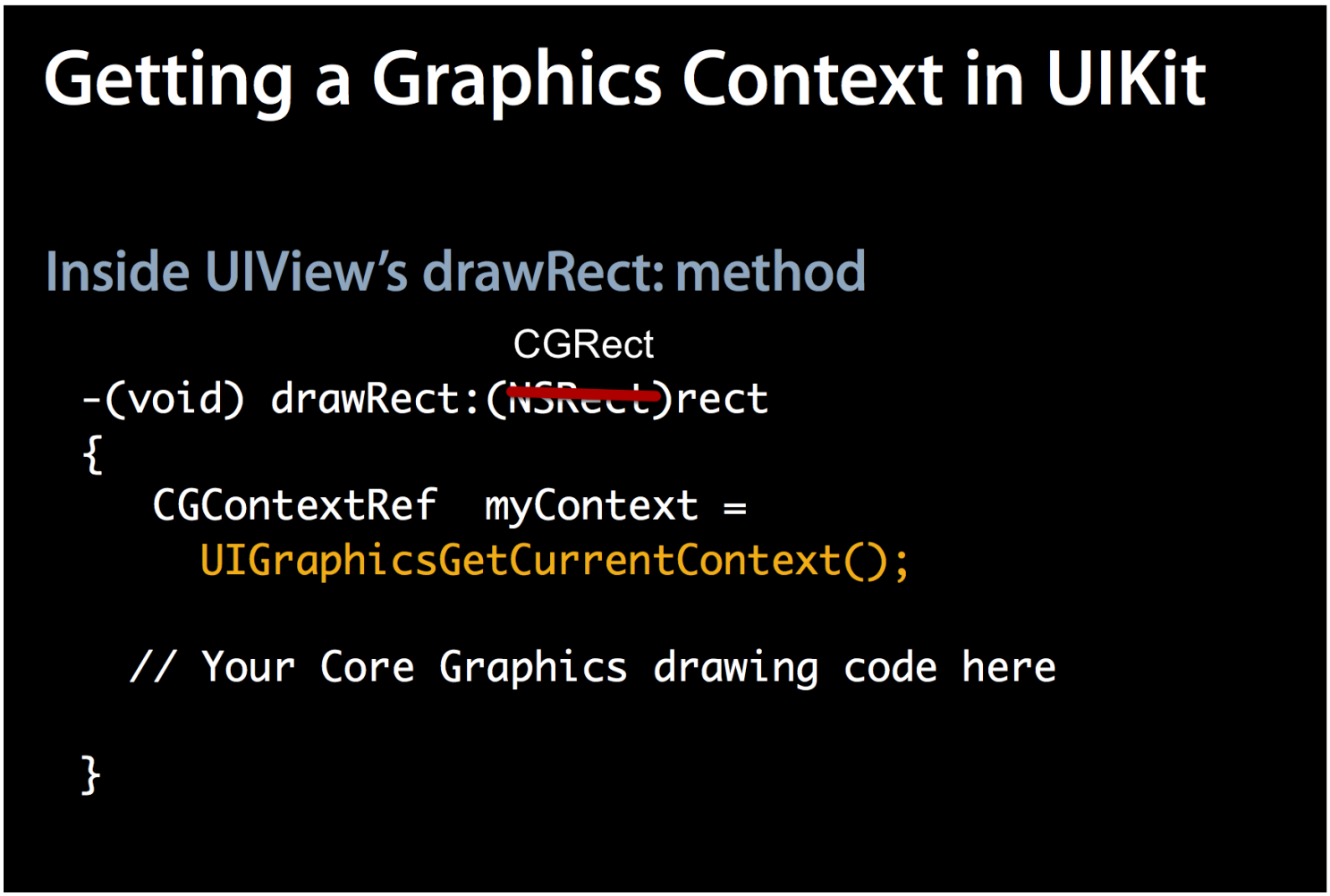
Operation	Function
Scale	CGContextScaleCTM
Rotate	CGContextRotateCTM
Translate	CGContextTranslateCTM

"CTM" = Coordinate Transformation Matrix

Current Graphics Contexts

- This is what you draw into
- UIImage, NSString (for drawing text), UIBezierPath (for drawing shapes), and UIColor
- If using drawRect: and UIGraphicsBeginImageContext you have to create your own Graphics Contexts.
- If you've been given a context, use UIGraphicsPushContext and Pop

Get a Graphics Context



Subclass UIView. There is no drawRect in UIViewController, If we are creating a PDF, then we use create a CGPDFDocumentRef

(3) Ways to get Context

- UIGraphicsBeginImageContexts - not only creates an image context, it also makes that context the current graphics context.
- When drawRect is called, the UIView's drawing context is already the current graphics context.
- Given a context: argument - is a reference to a graphics context (Push/Pop, Save/ Restore)

Drawing More Complex Shapes

- Common steps for drawRect: are
  - Get current graphics context
  - Define a path
  - Set a color
  - Stroke or fill path
  - Repeat, if necessary

Graphics State

Parameters that control drawing

- Stroke color
- Fill color
- Coordinate space transformation (CTM)
- Line width, join, cap, dash, miter limit
- Global alpha
- Shadow
- Blend mode

. . . and more!

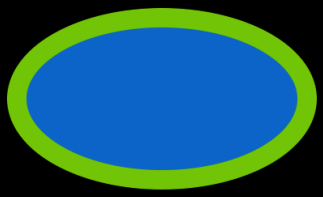
Filling and Stroking Functions

Rectangles

```
void CGContextFillRect (
    CGContextRef context,
    CGRect rect
);

void CGContextStrokeRect (
    CGContextRef context,
    CGRect rect
);
```

Ellipses

```
void
CGContextFillEllipseInRect (
    CGContextRef context,
    CGRect rect
);

void
CGContextStrokeEllipseInRect (
    CGContextRef context,
    CGRect rect
);
```



Methods

Filling and Stroking Rectangle w/Translate

A Filling and Stroking Routine

```
void doStrokedAndFilledRects(CGContextRef context)
{
 CGRect ourRect = CGRectMake(10, 10, 130, 100);

 CGContextSetFillColorWithColor(context, myGetBlueColor());
 CGContextFillRect(context, ourRect);
 CGContextSetStrokeColorWithColor(context, myGetGreenColor());
 CGContextStrokeRectWithWidth(context, ourRect, 10);

 CGContextSaveGState(context);
      CGContextTranslateCTM(context, 200, 0);
      CGContextStrokeRectWithWidth(context, ourRect, 10);
      CGContextFillRect(context, ourRect);
 CGContextRestoreGState(context);

 // Draw more rectangles here
}
```

Example of StrokeRect and Translation

Filling and Stroking Routine

Filling and Stroking

```
CGContextRef ctx =
UIGraphicsGetCurrentContext();
CGRect ourRect = CGRectMake(10,10,130,100);
CGContextSetFillColorWithColor(ctx, [UIColor
whiteColor].CGColor);
CGContextStrokeRectWithWidth(ctx, ourRect, 10);

CGContextSaveGState(ctx);
CGContextTranslateCTM(ctx,150,0);
CGContextStrokeRectWithWidth(ctx,ourRect, 10);
CGContextFillRect(ctx, ourRect);
CGContextRestoreGState(ctx);
```

Filling and Stroking Rectangle (image)

CGContextDrawPath

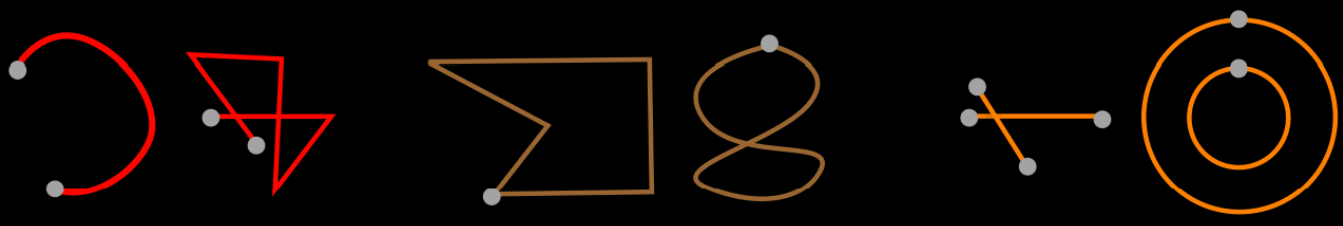
- Fills and Strokes with one Command
- Otherwise use CGContextFill... or CGContextStroke...

CGPath

- Two parallel sets of functions for using paths
  - CGContext "convenience" throwaway functions
  - CGPath functions for creating reusable paths

CGContext	CGPath
CGContextMoveToPoint	CGPathMoveToPoint
CGContextLineToPoint	CGPathAddLineToPoint
CGContextAddArcToPoint	CGPathAddArcToPoint
CGContextClosePath	CGPathCloseSubPath
And so on and so on . . .	

Creating Core Graphics Paths



Open Paths | **Closed Paths** | **Multiple Subpaths** (open and closed)

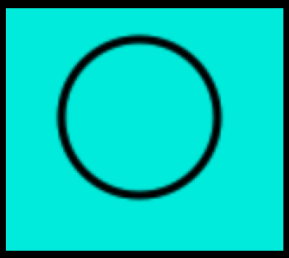
- Begin path
- Add primitives to the path
  - Line segments, Bézier cubic and quadratic curves
- Add entire shapes to the path
  - Rectangles, ovals, arcs
- Close path (optional)
- Paint the path

Paths

- CoreGraphics paths define shapes
- Made up of lines, arcs, curves and rectangles
- Creation and drawing of paths are two distinct operations
  - Define path first, then draw it

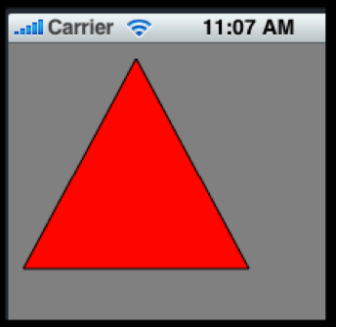
Circle

```
CGContextSetFillColorWithColor(ctx, [UIColor
blueColor].CGColor);
        CGContextBeginPath(ctx);
        CGContextSetLineWidth(ctx,3.0);
        CGContextAddArc (ctx, 150.0, 50.0, 30.0, 0.0,
2.0 * 3.142,YES);
        CGContextStrokePath(ctx); // or
        CGContextDrawPath(ctx,kCGPathFillStroke);
```



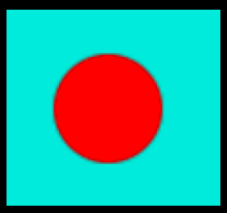
Simple Path Example

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

    [[UIColor grayColor] set];
    UIRectFill ([self bounds]);

    CGContextBeginPath (context);
    CGContextMoveToPoint (context, 75, 10);
    CGContextAddLineToPoint (context, 10, 150);
    CGContextAddLineToPoint (context, 160, 150);
    CGContextClosePath (context);

    [[UIColor redColor] setFill];
    [[UIColor blackColor] setStroke];
    CGContextDrawPath (context, kCGPathFillStroke);
}
```



Circle

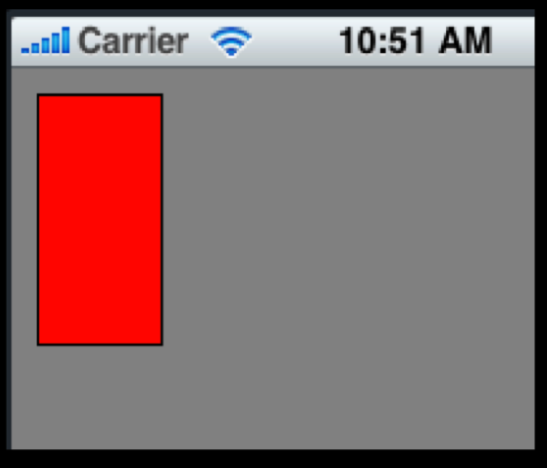
```
CGMutablePathRef path =
CGPathCreateMutable();

CGPathAddArc(path, NULL, 400.0, 75.0, 20.0, 0.0,
2.0 * 3.142,YES);

CGContextSetFillColorWithColor(ctx, [UIColor
redColor].CGColor);

CGContextAddPath(ctx, path);

CGContextFillPath(ctx);

CGPathRelease(path);
```



Simple drawRect: example

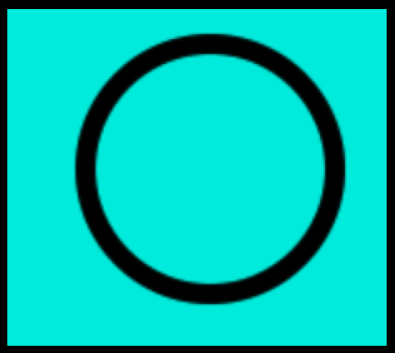
- Draw a solid color and shape

```
- (void)drawRect:(CGRect)rect {
    CGRect bounds = [self bounds];

    [[UIColor grayColor] set];
    UIRectFill (bounds);

    CGRect square = CGRectMake (10, 10, 50, 100);
    [[UIColor redColor] set];
    UIRectFill (square);

    [[UIColor blackColor] set];
    UIRectFrame (square);
}
```



What happened to the idea of getting a context?

Circle CG



```
CGContextSetLineWidth(ctx, 8.0);

CGRect rectangle = CGRectMake(250, 50,
100, 100);

 CGContextAddEllipseInRect(ctx, rectangle);

 CGContextStrokePath(ctx);
```

Blue Circle in CoreGraphics

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

CGContextAddEllipseInRect(ctx,CGRectMake(0,0,100,
100));
CGContextSetFillColorWithColor(con, [UIColor
blueColor].CGColor);

 CGContextFillPath(ctx);
}
```

Blue Circle

```
- (void) drawRect: (CGRect) rect {

UIBezierPath* p = [UIBezierPath
bezierPathWithOvalInRect:CGRectMake(0,0,
 100,100)];

 [[UIColor blueColor] setFill];

 [p fill];

}
```

Create Image* Core Graphics

```
- (void) drawRect: (CGRect) rect {

        UIGraphicsBeginImageContextWithOptions(CGSizeMake(100,100)
        , NO, 0);

        CGContextRef con = UIGraphicsGetCurrentContext();

        CGContextAddEllipseInRect(con, CGRectMake(0,0,100,100));

        CGContextSetFillColorWithColor(con, [UIColor
        blueColor].CGColor);

        CGContextFillPath(con);

        UIImage* im = UIGraphicsGetImageFromCurrentImageContext();

        UIGraphicsEndImageContext();

}
```

Create Image* UIKit

```
UIGraphicsBeginImageContextWithOptions(CGSizeMake(100,100), NO, 0);

UIBezierPath* p = [UIBezierPath bezierPathWithOvalInRect:CGRectMake(0,0,100,100)];

[[UIColor blueColor] setFill];

[p fill];

UIImage* im = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();
```

No Current Graphic Required

Coordinates (ULO, LLO)

A word on Coordinates

- If Using UIKit, then drawing into CurrentGrapics works as you'd expect (ULO)
- If Using CGraphics, any ~~UIKit~~ CGImage drawing (e.g. CGImageRef) will draw upside down (LLO)

Creating PDF Document

Drawing PDF Documents

- CGPDFDocumentRef represents a PDF file
- CGPDFPageRef represents a page in a PDF file
  - Pages start at 1, not 0

PDF Document Functions

```
// Create a PDF document from a URL
CGPDFDocumentRef CGPDFDocumentCreateWithURL(CFURLRef url);

// Return the number of pages in the document
size_t CGPDFDocumentGetNumberOfPages(CGPDFDocumentRef
document);

// Return a specific page in the document
CGPDFPageRef CGPDFDocumentGetPage(CGPDFDocumentRef document,
    size_t pageNumber);

// Draw a page in a context
void CGContextDrawPDFPage(CGContextRef c, CGPDFPageRef page);
```

Capturing Screen Gestures

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

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
    [self.myPath addLineToPoint:point]; // Adds and Draws CGPoints to Screen
       [self setNeedsDisplay];
}
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
       [self processGestureData];  // Does “whatever” after done tracing
}
```

Draw BezierPath using CG

```
- (void)drawRect:(CGRect)rect
{
        // Drawing code, get context
        CGContextRef ctx = UIGraphicsGetCurrentContext();

        // settings
        CGContextSetStrokeColorWithColor(ctx, [UIColor
grayColor].CGColor);

// Draw
[_myPath strokeWithBlendMode:kCGBlendModeNormal
alpha:1.0]; }
```

How to Recognize Shapes

http://faculty.washington.edu/wobbrock/pubs/uist-07.1.pdf

Geometric Template Matching ($1)

- Step 1: Resample the Point Path
- Step 2: Rotate Once Based on the "Indicative Angle"
- Step 3: Scale and Translate
- Step 4: Find the Optimal Angle for the Best Score

Objective-C $1 Implementation

http://giraffelab.com/code/GLGestureRecognizer/

More Drawing Information

- UIView Class Reference
- CGContext Reference
- “Quartz 2D Programming Guide”
- Lots of samples in the iPhone Dev Center