CS193E
Lecture 9

Views and Drawing

Today's Topics

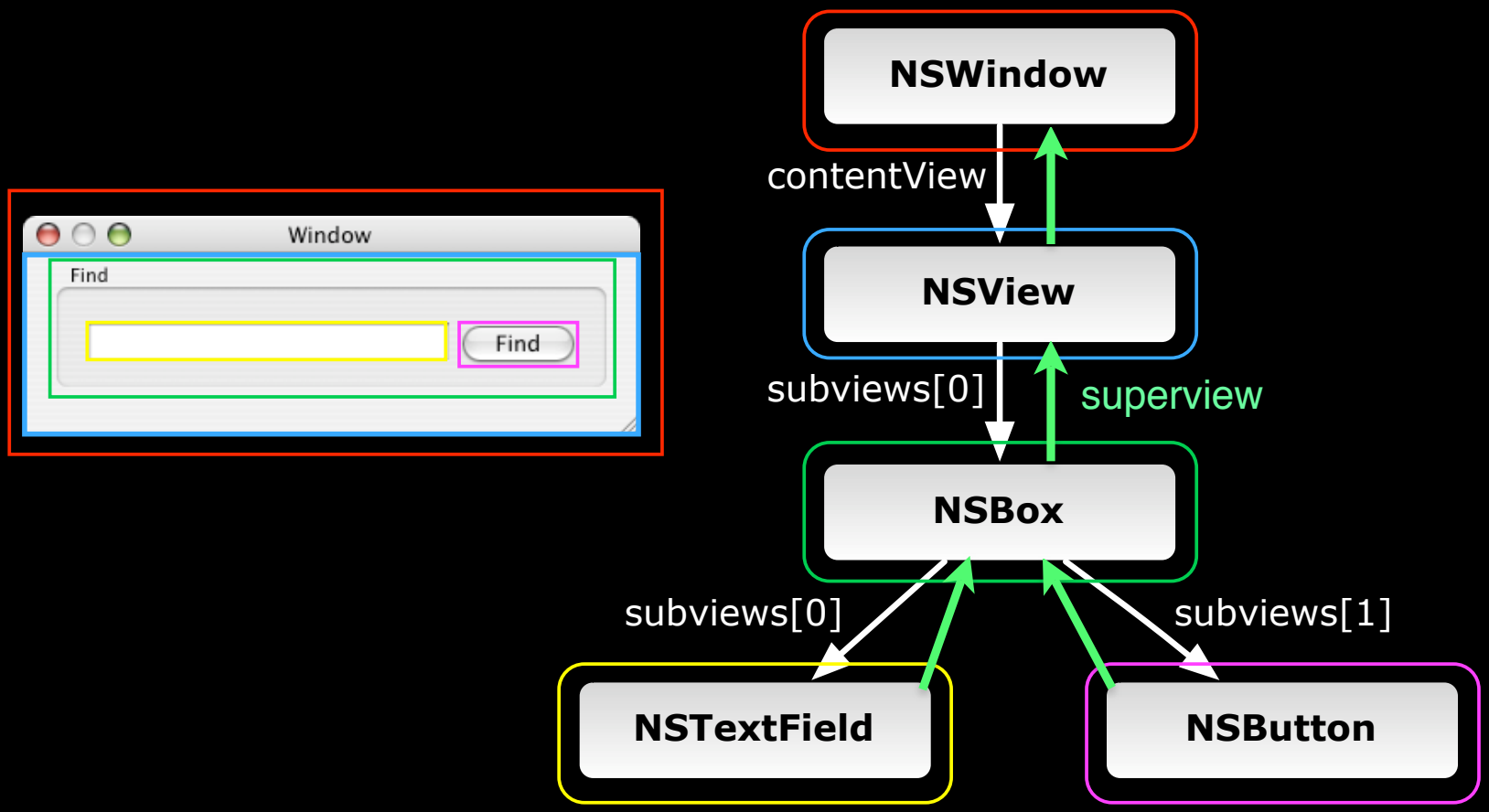
- Questions?
- The View Hierarchy
- Drawing in Cocoa
- Drawing Examples

The View Hierarchy

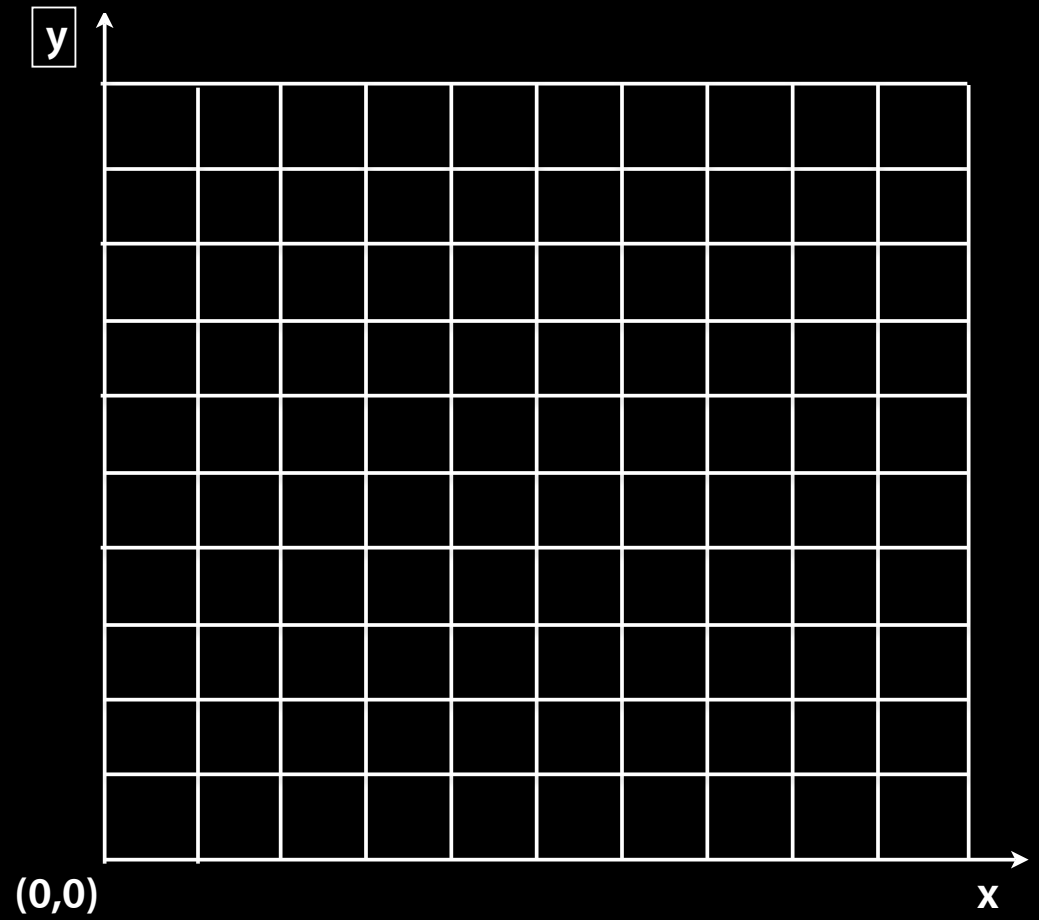
What is a View?

- NSView manages a ‘rectangle’ in a window
- Input: Handles events directed at it
- Output: Draws its contents
- Contains subviews which are stacked above it
- A view’s parent is its “superview”

View Hierarchy



View Coordinate System



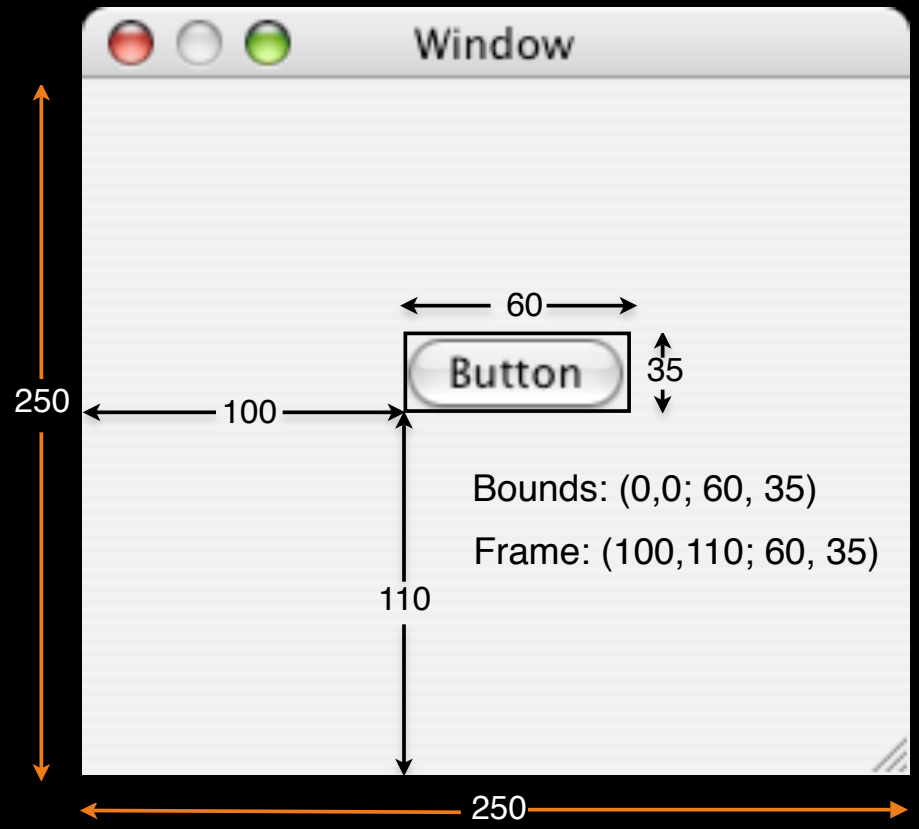
View Coordinate System

- Device-independent
- 2-D floating-point Cartesian coordinates
- Origin at lower-left corner
- One unit = 1/72nd of an inch (kind of)
- Coordinate system can be translated, scaled, and rotated

Bounds vs. Frame

- A view's dimensions are represented by its "bounds" rectangle
  - The bounds origin is almost always (0, 0)
- The dimensions of a view in the coordinate system of its superview is its "frame" rectangle

Bounds vs. Frame



Bounds: (0,0; 250, 250)

Frame: (0, 0; 250, 250)

Drawing

NSView Drawing Model

- Drawing code is placed in the drawRect: method of your NSView subclass

```
- (void)drawRect:(NSRect)rect
{
    /* Your drawing code here */
}
```

- You draw the content that should appear in “rect”
- Cocoa manages redrawing for you

NSView Drawing Model

- If a view needs to be redrawn, use

```
[view setNeedsDisplay:YES]
```

or

```
[view setNeedsDisplayInRect:dirtyRect]
```

- Dirty rects are batched up and redrawn at the end of every event

View initialization

- Designated initializer

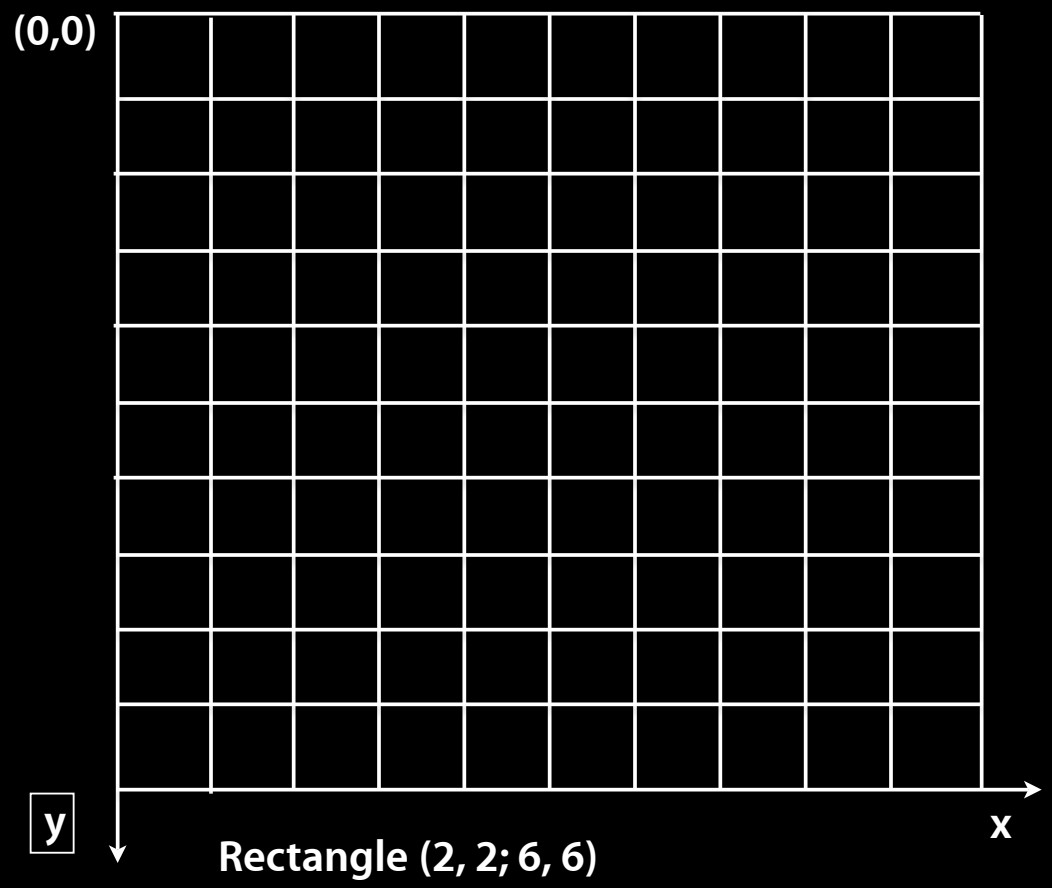
```
-(id)initWithFrame:(NSRect)frame
```

- When loading a nib file, to do work after all outlets are assigned

```
-(void)awakeFromNib
```

-isFlipped

NSView subclasses can override and return YES



Drawing Utilities

NSColor

- Built-in colors (very useful)
  - redColor, blueColor, and so on
- With a color space (for color-savvy people)
  - colorWithCalibratedRed:green:blue:alpha:
- Support for transparency (alpha)
- Set a color for drawing in a view with “set”
  - [[NSColor blackColor] set];

NSBezierPath

- A collection of Bézier paths
  - A Bézier path is a straight or curved line segment
- Used as the source for many operations
  - Filling
  - Stroking (outline)
  - Clipping (restricting drawing to a particular region)
  - Hit testing

NSBezierPath Conveniences

```
[NSBezierPath fillRect:rect];

[NSBezierPath strokeRect:rect];

[NSBezierPath strokeLineFromPoint:p1 toPoint:p2];
```

Color & Path Example

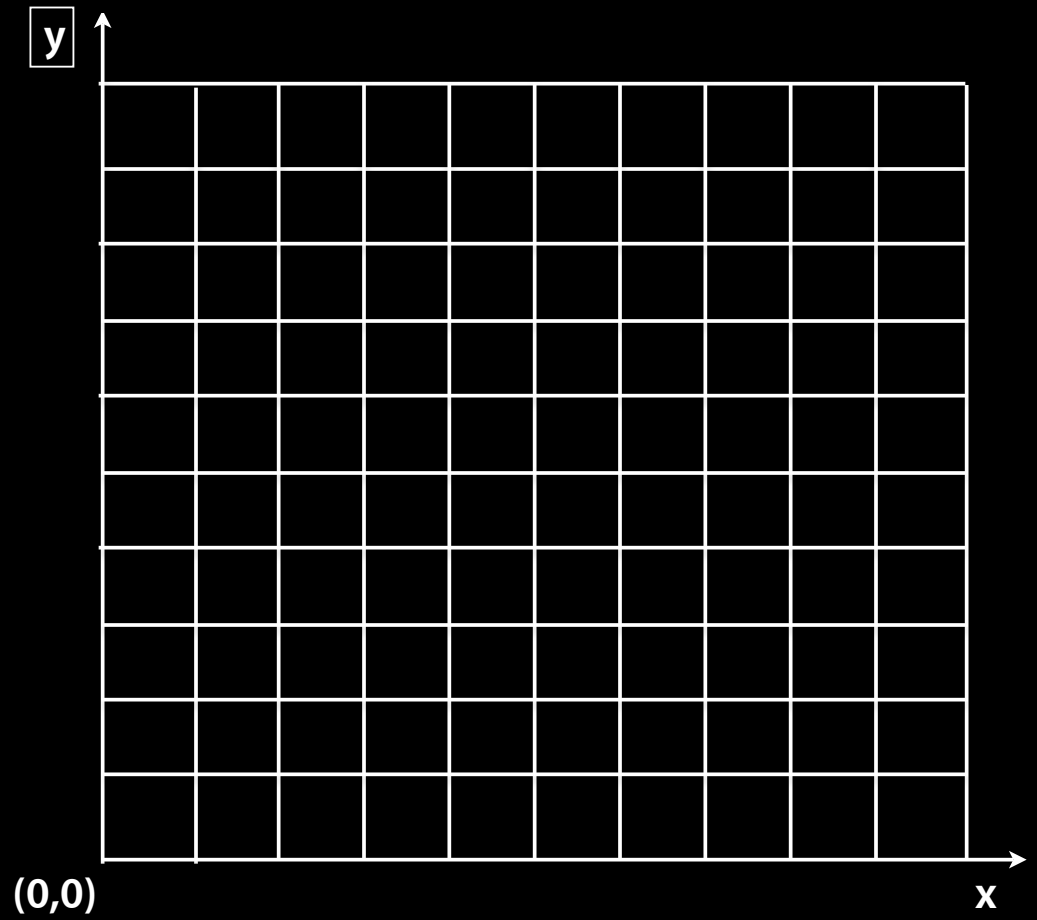
```
NSColor *red = [NSColor redColor];
[red set];

[NSBezierPath fillRect:[self bounds]];
```

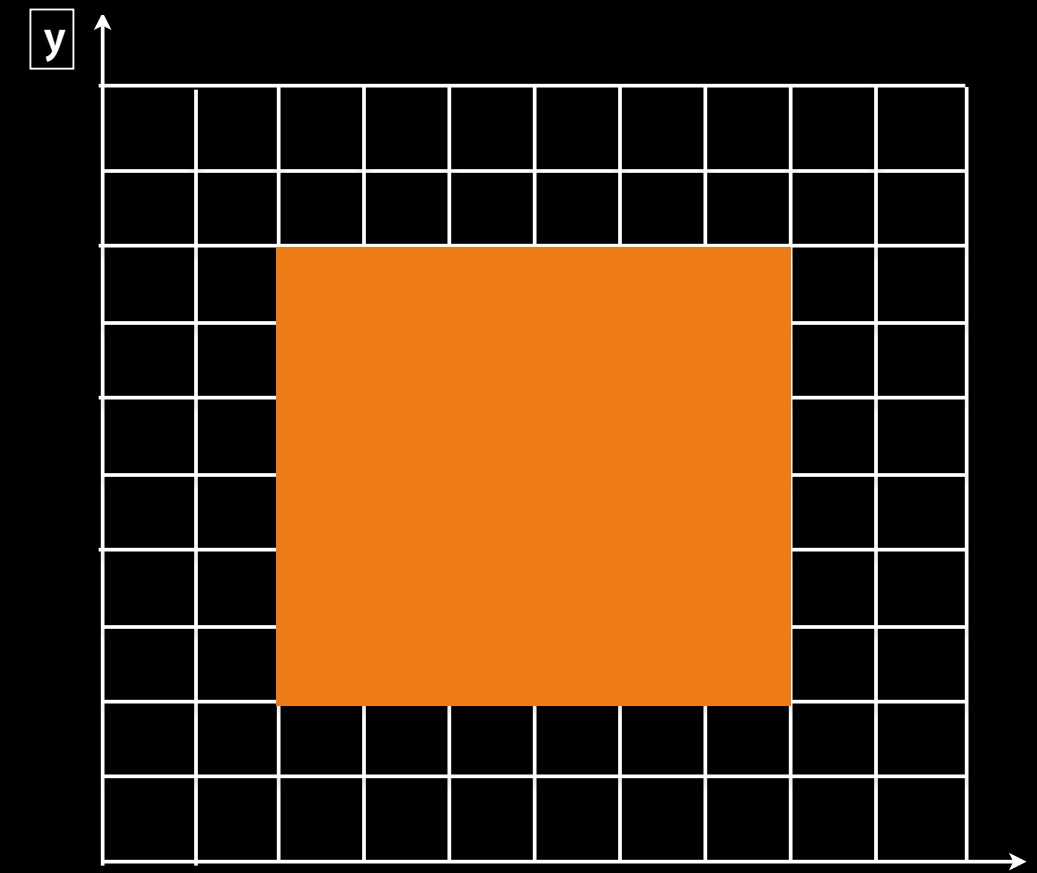
Color & Path Example

```
NSColor *pink = [NSColor colorWithCalibratedRed:1.0
    green:0.81 blue:0.85 alpha:1,0 ];
[pink set];
[NSBezierPath strokeRect:[self bounds]];
```

Drawing on pixels...

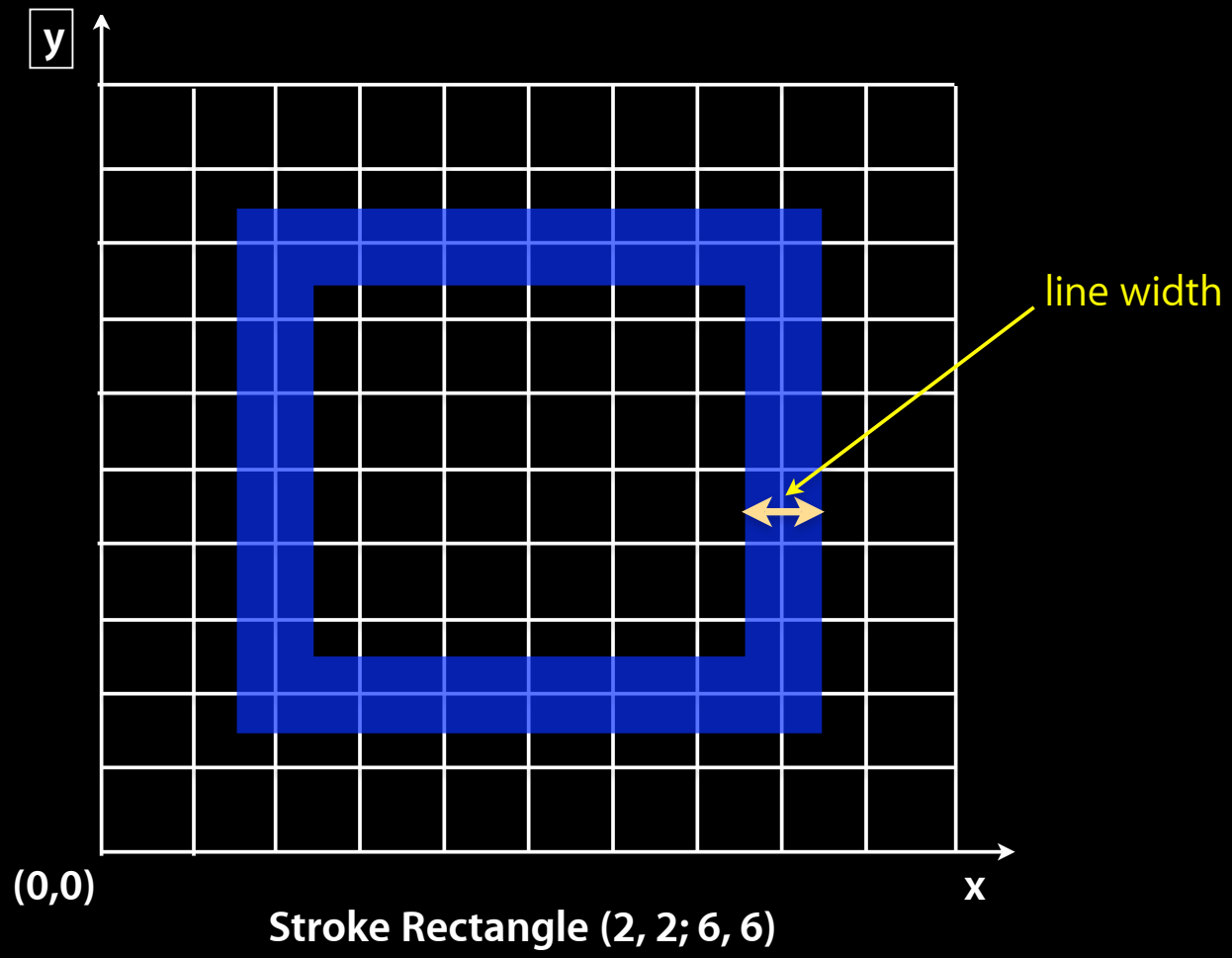


Fill a rectangle



Filled Rectangle (2, 2; 6, 6)

Stroke a rectangle



Drawing Text

```
NSFont *font = [NSFont fontWithName:@"Helvetica" size:24];

[dict setObject:font forKey:NSFontAttributeName];

[@"Hello World" drawAtPoint:point withAttributes:dict];
```

Text Attributes

NSFontAttributeName

NSForegroundColorAttributeName

NSBackgroundColorAttributeName

NSUnderlineStyleAttributeName

NSShadowAttributeName

NS$^{\text{Superscript}}$AttributeName

... and many others

Primitive Types

NSPoint

```
#import <Foundation/NSGeometry.h>

typedef struct _NSPoint {
    float x;
    float y;
} NSPoint;

NSPoint point = { 10.0, 20.0 };
NSPoint point = NSMakePoint(10.0, 20.0);
```

NSSize

```
#import <Foundation/NSGeometry.h>

typedef struct _NSSize {
    float width;
    float height;
} NSSize;

NSSize size = { 10.0, 20.0 };
NSSize size = NSMakeSize(10.0, 20.0);
```

NSRect

```
#import <Foundation/NSGeometry.h>

typedef struct _NSRect {
    NSPoint origin;
    NSSize size;
} NSRect;

NSRect rect = { { 10.0, 20.0 }, { 30.0, 30.0 } };
NSRect rect = NSMakeRect(10, 20, 30, 30);
```

NSRect Functions

NSMinX(rect)
NSMinY(rect)

NSMaxX(rect)
NSMaxY(rect)

NSEqualRects(rect1,rect2)
NSIsEmptyRect(rect)
NSUnionRect(rect1,rect2)
NSIntersectionRect(rect1,rect2)
NSPointInRect(point, rect)
NSInsetRect(rect, deltaX, deltaY)

Demo

Custom Views in IB

Questions?