Mobile Application Development
Aileen Pierce

ANIMATIONS

Animations

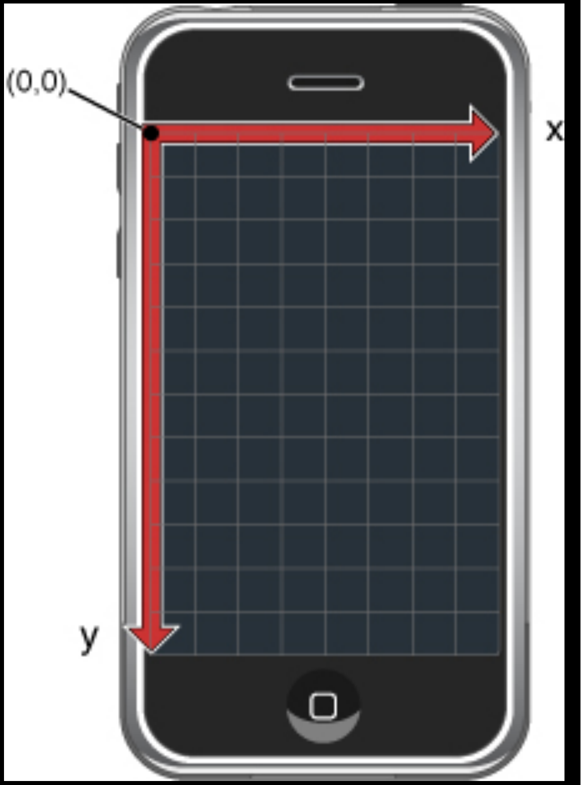
- Use the **`NSTimer`** class to animate views
- Core Graphics framework
- Animate UIViews
  - Translation
  - Rotation
  - Scaling
- Core Animation for more advanced animation

NSTimer

- The **`NSTimer`** class creates timer objects which can call a method at a regular interval
- The **`scheduledTimerWithTimeInterval(_, target, selector, userInfo, repeats)`** method creates a new timer with an interval
- After an **`NSTimer`** object is started you cannot change its firing interval. You must stop the timer and create a new one.
- The **`invalidate()`** method stops the timer

Core Graphics Framework

- Core Graphics provides the basic building blocks for drawing and positioning views.
- Core Graphics is a C API (not object-oriented)
- The iOS coordinate system is set up so that point (0,0) is in the top-left corner.
  - Y values grow larger going down the screen
  - X values grow larger going across the screen
- Units are points not pixels



Core Graphics

- CGPoint ={x,y} defines a point
  - center property defines the center point
- CGSize={width, height} defines the size
- CGRect ={origin x, origin y, size width, size height} describes the rectangle a UIView lives in
- There are built-in macros for creating each of these

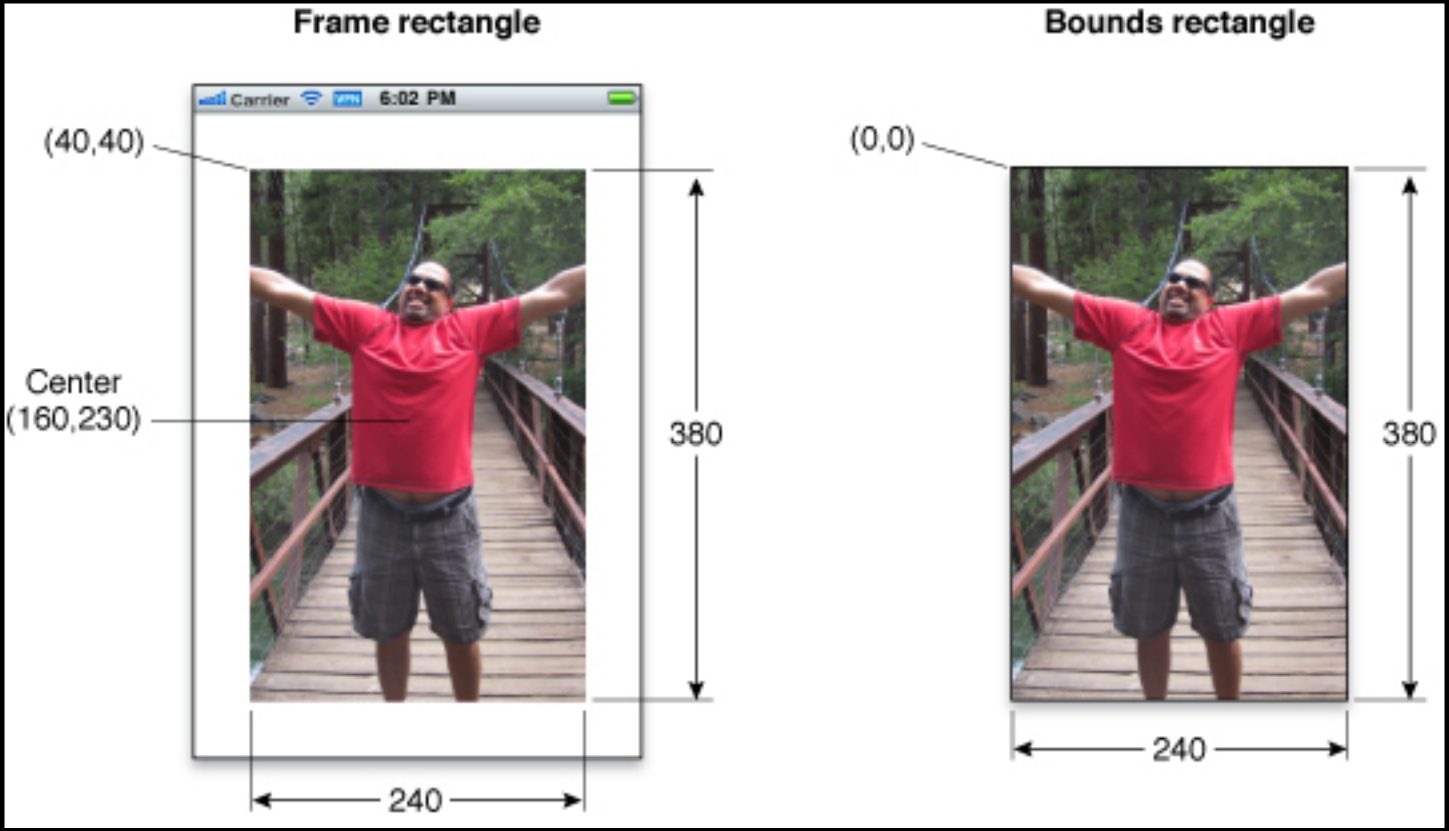
Core Graphics

- CGPoint contains x and y coordinates
  - var p = CGPointMake(34.5, 22.0)
  - p.x+=20 move right by 20 points
- CGSize contains width and height
  - var s = CGSizeMake(100.0, 200.0)
  - s.height+=50 make the size 50 points taller
- CGRect contains the origin CGPoint and CGSize
  - var r= CGRectMake(45.0, 75.5, 300, 500)
  - r.size.height+=45 make the rectangle 45 points taller
  - r.origin.x += 30 move the rectangle to the right 30 points

Core Graphics

- A UIView object tracks its size and location using its frame, bounds, and center properties
- Frame: a rectangle positioned from the perspective of the parent view
- Bounds: a rectangle positioned from the perspective of the view itself, usually at (0,0)
- Center: the center of your view in the frame
- To move a view you need to either set its center or frame property.

Core Graphics



Animation

- Animate UIViews with animation closures
- Animation closures define how a view animates
- Start an animation by calling **`beginAnimations(_, context:)`**
- An animation ends by calling **`commitAnimations()`**
- All animated objects must be UIView subclasses that are part of the view hierarchy

Animation Closures

- A closure is a group of executable code like a function
- The closure body is in {}
- Animation methods
  - **`animateWithDuration(_, animations)`**
  - **`animateWithDuration(_, animations, completion)`**
  - **`animateWithDuration(_, delay, options, animations, completion)`**
- The **`animations`** and **`completion`** parameters take closures

Animation Properties

- Animatable properties in UIView animation closures
  - Frame
  - Bounds
  - Center
  - Transform – scale, rotation, offset
  - Alpha – view's transparency (default is 1 not transparent)
  - backgroundColor

Animation Customization

- There are other methods that customize animations
  - Delay start time
  - Start at a specific time
  - Curve
  - Duration
  - Repetition
  - Autoreverse

Animation Curves

- **`UIViewAnimationOptions`** indicate how you want to perform the animation
  - **`CurveEaseInOut`**
    - Start slow, speed up, end slow
    - Default animation curve
  - **`CurveEaseIn`**
    - Start slow, end fast
  - **`CurveEaseOut`**
    - Start fast, end slow
  - **`CurveLinear`**
    - Keeps the same speed throughout the animation

Transformations

- A view's transform property is of type CGAffineTransform and allows you to rotate, scale, and transform your view.
  - **`CGAffineTransformMakeTranslation(_, _)`** returns a transform you can use to move your view
  - **`CGAffineTransformMakeRotation(_)`** returns a transform you can use to rotate your view
  - **`CGAffineTransformMakeScale(_, _)`** returns a transform you can use to scale your view