Core Animation in Practice

Part 1

Michael Levy
Senior Engineer, Advanced Quartz Technologies

Core Animation in Practice

What will we cover?

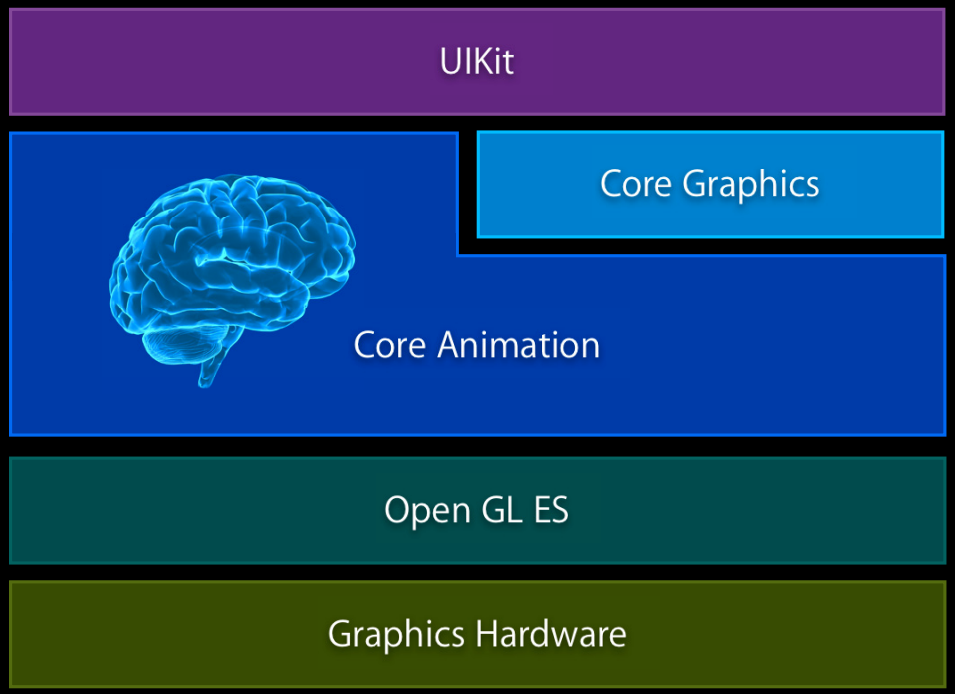
- Layers and their properties
- Providing layer content
  - Core Graphics
  - UIKit
  - OpenGL ES
  - AVPlayer
- Animating layer properties

Core Animation

Architecture

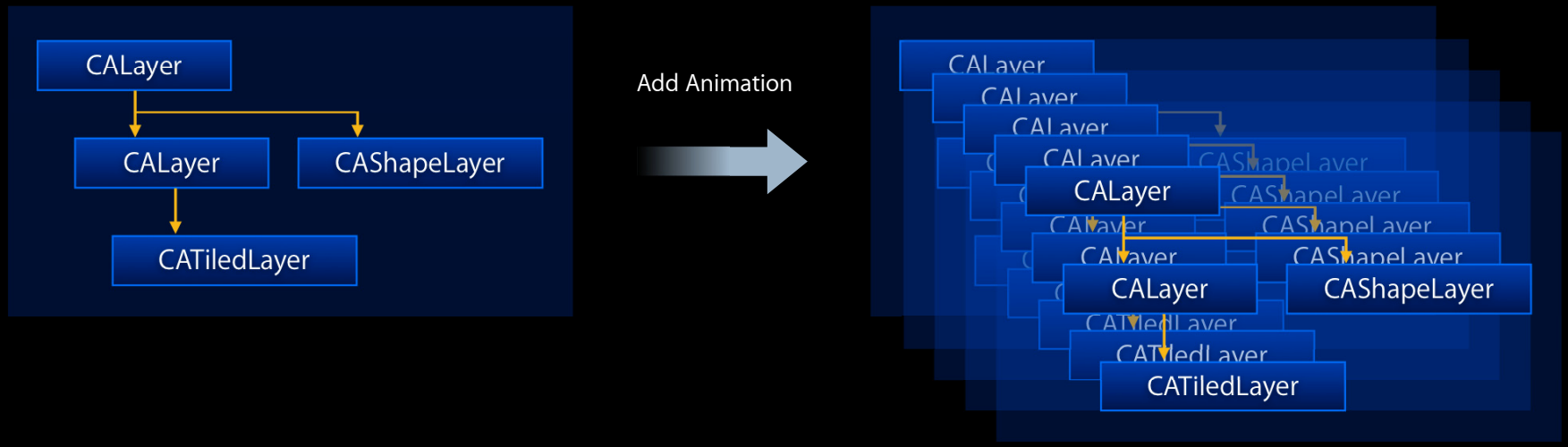
Core Animation

Architecture



Core Animation Architecture

Rendering



UIKit and Core Animation

Which should you use?

- Almost always…UIKit
- Core Animation?
  - Lightweight
  - Short-lived
- Benefits of understanding Core Animation
  - Improve your effectiveness with UIKit animation
  - Improve your app's performance where necessary

Animation Basics

Layers and their properties

Core Animation

The magic sauce

animation [ˌænɪˈmeɪʃən]
n. the state of being alive
(source: Concise Oxford Dictionary)

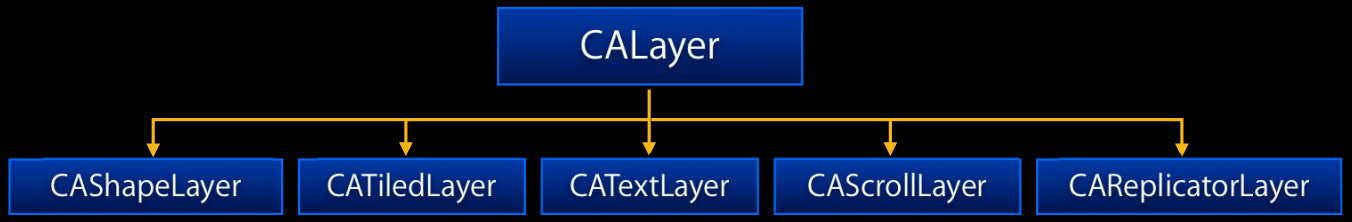
declarative [d-klâr-tv, -klr-]
adj. form of to declare:
announce formally

Layers

Animatable properties

Declarative model

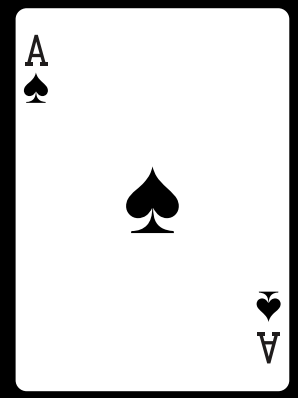
Layers

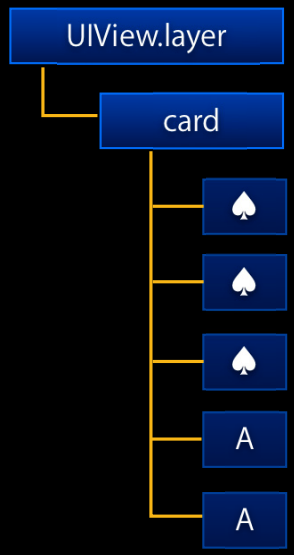

CALayer
CAShapeLayer
CATiledLayer
CATextLayer
CAScrollLayer
CAReplicatorLayer

Layers
Creating a layer

- Framework is QuartzCore
  - Not included in template projects

```
#import <QuartzCore/QuartzCore.h>
...
CALayer* myLayer = [CALayer layer];
myLayer.bounds = CGRectMake(0,0,w,h);
myLayer.position = CGPointMake(30.0,67.0);
myLayer.content = caLogo;
[self.layer addSubLayer:myLayer];
```





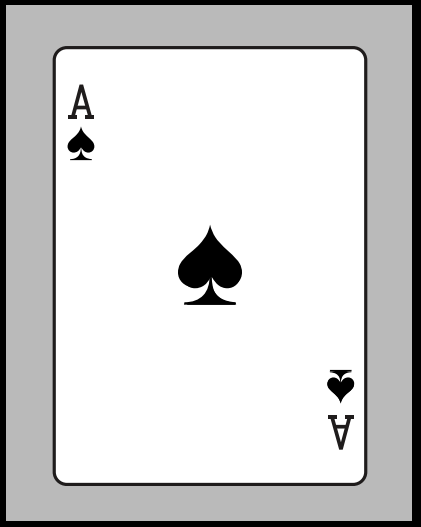
Layers

Sublayers and layout

- Model similar to UIView
  - addSubLayer:
  - insertSubLayer:above: (etc.)
  - setNeedsLayout
  - setNeedsDisplay
  - layoutSubLayers
  - …
- 2 1/2 D model
  - Transform is 3D

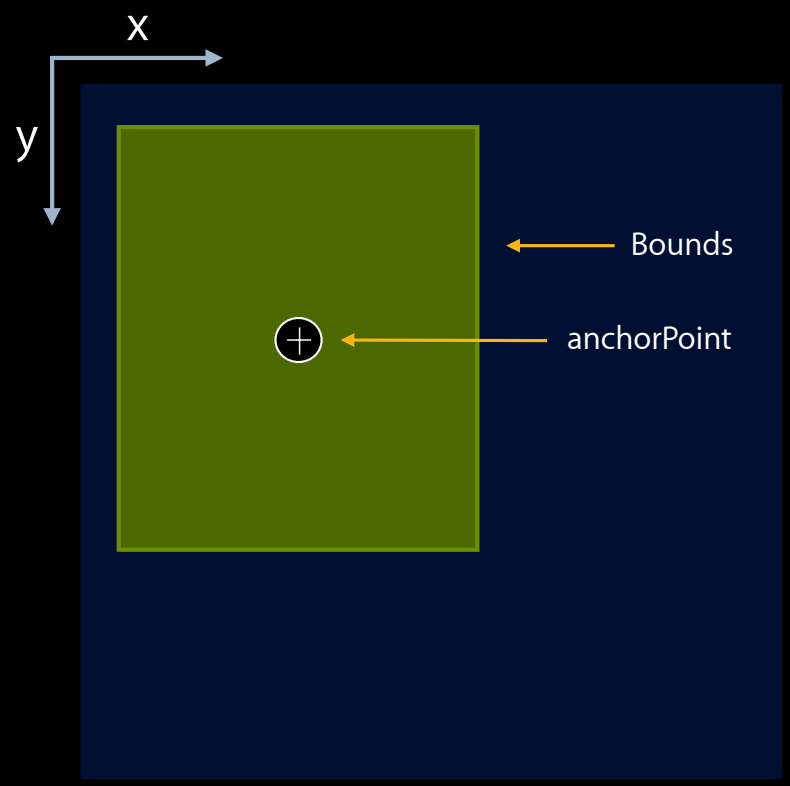
Declarative Style

```
spadeAce = [CALayer layer];
...(various properties)
// Add to view's layer
[self.layer addSublayer:spadeAce];
// Add the pips
// Center
CAShapeLayer* centerPip = [Cards spadePip];
centerPip.position = CGPointMake(...,...);
[spadeAce addSubLayer:centerPip];
// Bottom
CAShapeLayer* bottomPip = [Cards spadePip];
CATransform3D transform = CATransform3DMakeScale(0.5, 0.5, 1);
transform = CATransform3DRotate(transform, M_PI, 0, 0, 1);
bottomPip.transform =transform;
bottomPip.position = ...
[spadeAce addSublayer:bottomPip];
...
```

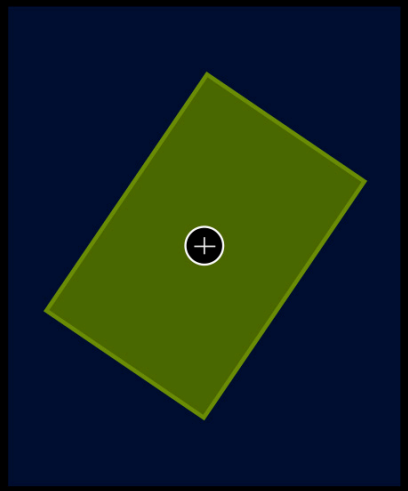


Layers

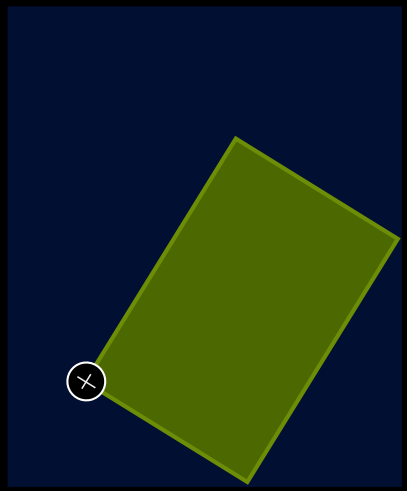
Layer geometry




```
bounds—CGRect
position—CGPoint (superlayer coordinates)
anchorPoint—CGPoint
transform—CATransform3D
```



```
// A: Rotate about center
layer.anchorPoint = CGPointMake(0.5,0.5);
layer.transform = rotationTransform;
```



```
 // B: Rotate about lower left
layer.anchorPoint = CGPointMake(0.0,1.0);
layer.transform = rotationTransform;
```

Demo

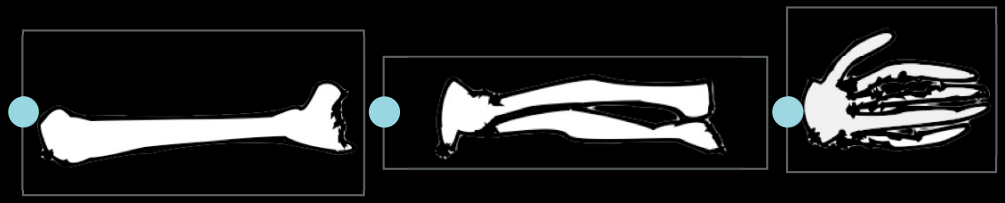
Humerous

↓

UlnaRadius

↓

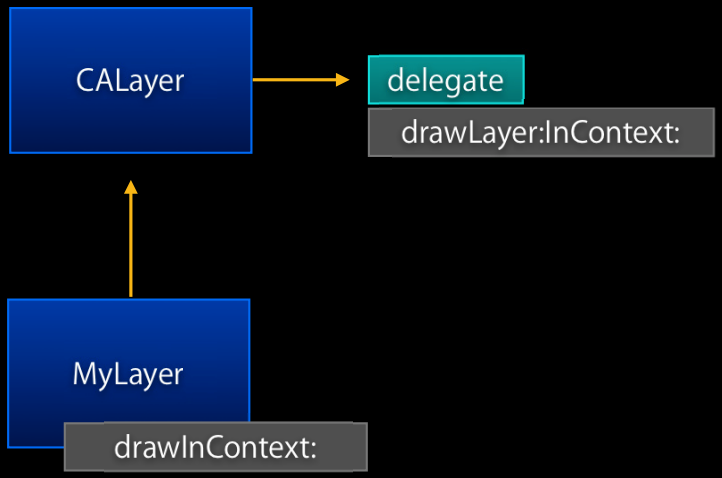
Hand



Providing Layer Content

Layers

Providing layer content



- Set delegate and implement drawLayer:InContext:
- Subclass and implement drawInContext:
- Context is a CGContextRef

Layer Content

Core Graphics

- Concise and elegant model
- Full 2D graphic capabilities
  - Comprehensive color support
  - Resolution independence
  - Device independence

Core Graphics

The essential concepts you need to know

- Abstract 2D coordinate space
- Unit is CGFloat
  - Point not necessarily a pixel
  - Origin top-left or bottom-left
- CGContextRef
  - Drawing destination
  - State (path, fill and stroke colors, line width,…)

Core Graphics

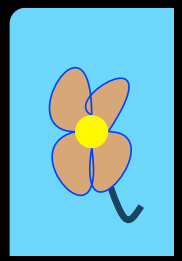
API model

- C-based
- Usual reference-based memory model
- CFType compliant

```
CGColorRef stunningColor = CGColorCreate(rgbColorSpace,rgbComponents);
CGContextSetFillColorWithColor(context,stunningColor);
CGColorRelease(stunningColor);
```

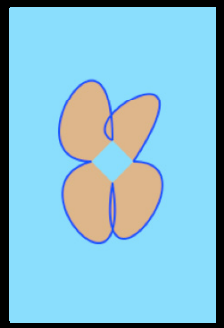
Core Graphics

Small example



1/4 of our card

- Back of a playing card
- Want two-way symmetry
- Use a stroked path



Example

Basic design

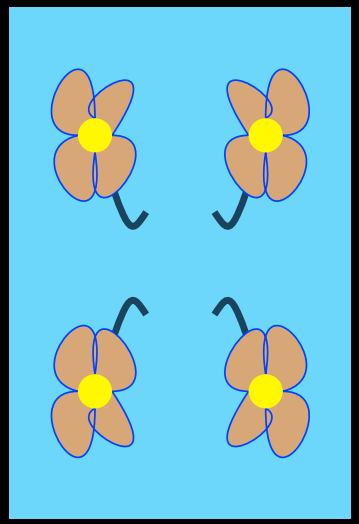
```
CGContextSaveGState(ctx);
...
CGMutablePathRef petal = CGPathCreateMutable();
CGPathMoveToPoint(petal, NULL, p1.x, p1.y);
CGPathAddCurveToPoint(petal, NULL, cp1.x, cp1.y, cp2.x, cp2.y, p2.x, p2.y);
CGPathMoveToPoint(petal, NULL, p2.x, p2.y);
CGPathAddCurveToPoint(petal, NULL, cp3.x, cp3.y, cp4.x, cp4.y, p3.x, p3.y);
...
CGContextAddPath(ctx, petal);
CGContextFillPath(ctx);
CGContextAddPath(ctx, petal);
CGContextStrokePath(ctx);
...
CGPathRelease(petal);
CGContextRestoreGState(ctx);
```

Core Graphics
Transformation

- Uses 2D Affine Transforms
  - Scale
  - Rotate
  - Translate
- Transform the coordinate space, not objects

Core Graphics

```
CGContextSaveGState(ctx);
[self drawDesign:ctx]
CGContextScaleCTM(ctx, 1, -1);
CGContextTranslateCTM(ctx, 0, 2*r.size.height);
[self drawDesign:ctx];
// Two similar blocks of code
....
CGContextRestoreGState(ctx);
```



Core Graphics

Other parts of Core Graphics

- Anti-aliasing
- Gradient fill
- Shading
- Patterns
- Line properties

CoreGraphics

Tip: Use UIKit!

- CG objects often wrapped in UIKit objects
- Available as properties
  - [UIColor redColor].CGColor
  - myUIImage.CGImage
- Useful to let UIKit do heavy lifting, and then you pick the CG object

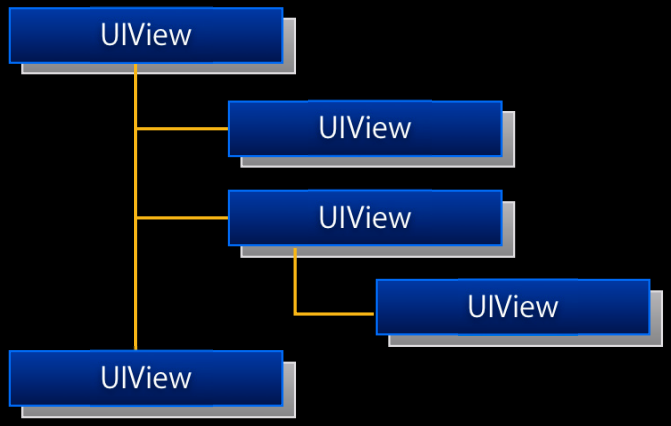
Other Ways to Create Layer Content

Open GL ES and Layers

AVPlayer Layer

Providing Content

UIView

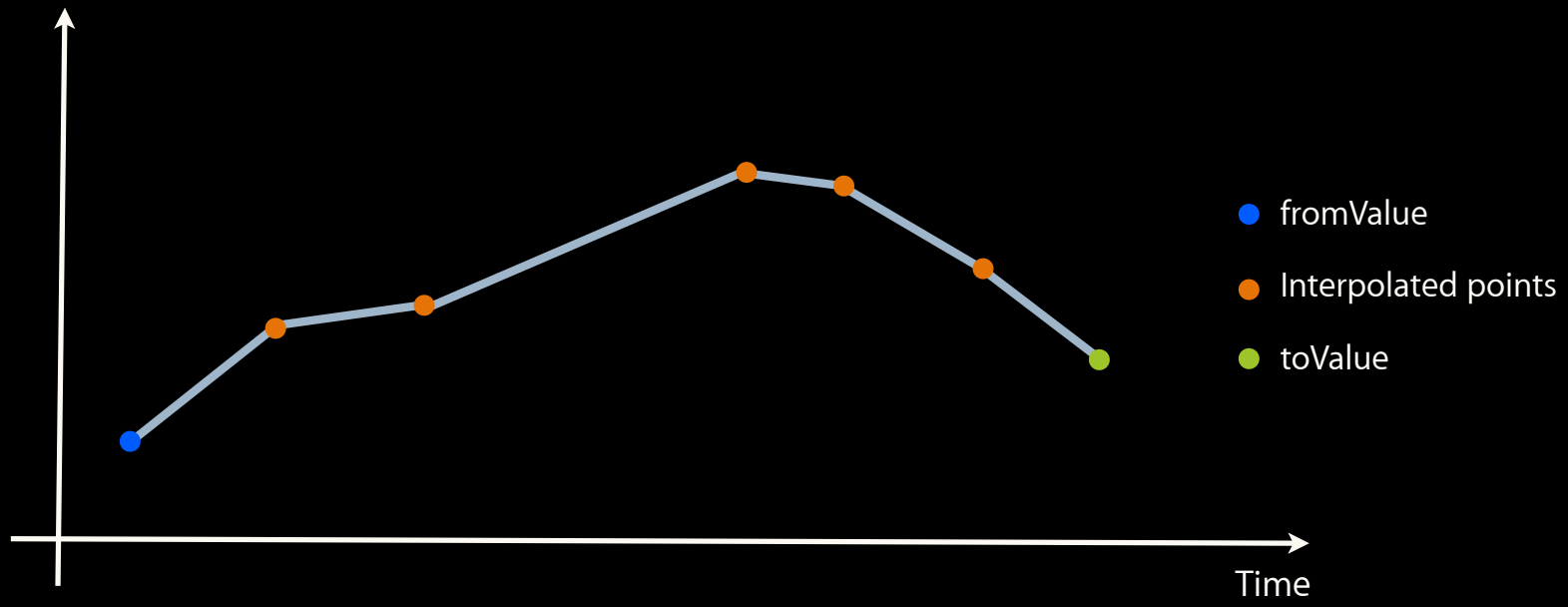


- Every UIView is backed by a layer
- Default is CALayer
  - Overide by implementing
  - +(Class) layerClass;
    - One of the CALayer subclasses
    - Your own subclass

Animation

Animation

What does animation mean?



Animation

Implicit animation

```
myLayer.position = nearBottomRight;
```

Animation

Implicit animation

```
[CATransaction setAnimationDuration:5];
[CATransaction setAnimationTimingFunction:
    kCAMediaTimingFunctionEaseInEaseOut];
myLayer.position = nearBottomRight;
```

Animation

Animatable properties

- Position—Move the layer relative to its superlayer
- Opacity—Fade in or out
- Shadow properties
- Transform—In 3D space
- Bounds—Grow, shrink
- And more—Almost all layer properties are animatable

Animation

Implicit animation

- Transaction
  - All animations during next run-loop
- CATransaction class
  - Duration
  - Timing function
  - Implicit or explicit

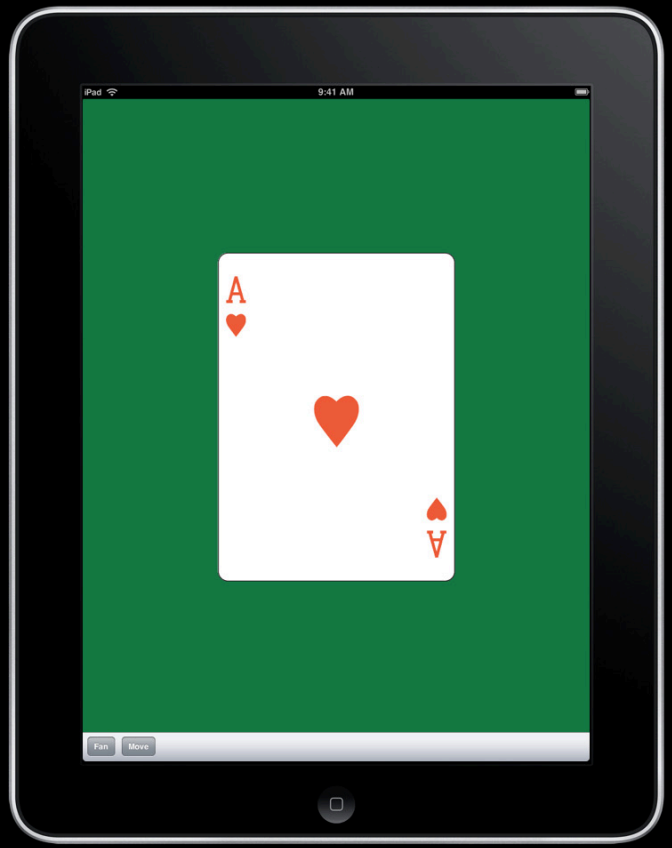
```
[CATransaction setDisableActions:YES]
```

Demo

Implicit Animation Example

```
for(k=0;k<3;++k)
{
  cards[k] = [CALayer layer];
  cards[k].bounds = b;
  cards[k].position = p;
  cards[k].anchorPoint = CGPointMake(0, 1);
  cards[k].contents = (id) images[k];
  [self.layer addSublayer:cards[k]];
}
```

Implicit Animation Example



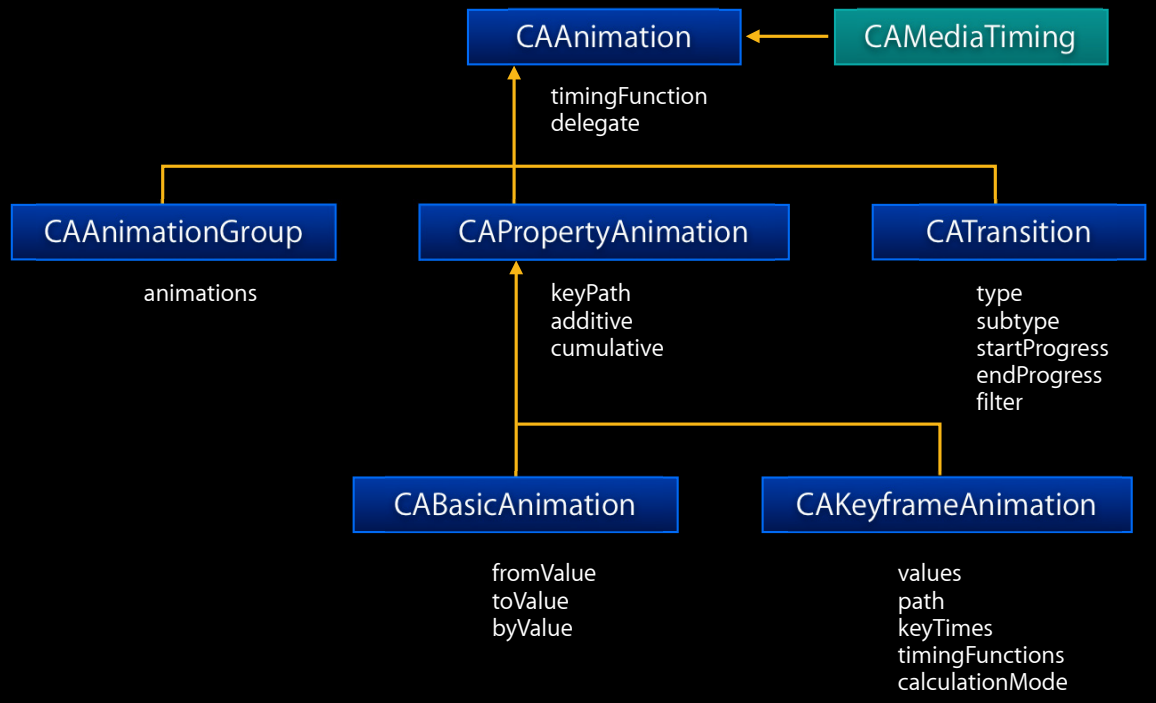
Implicit Animation Example

```
- (void) doAnimation:(id) sender
{
    CATransform3D r;
    r = CATransform3DMakeRotation(-0.3, 0, 0, 1);
    cards[0].transform = r;

    r = CATransform3DMakeRotation(0.0, 0, 0, 1);
    cards[1].transform = r;

    r = CATransform3DMakeRotation(0.3, 0, 0, 1);
    cards[2].transform = r;
}
```

Demo

Explicit Animation

Animation classes



Explicit Animations

Basic animations

- Which property?
  - Use keyPath
    - @"position"
    - @"position.y"
    - @"anchorPoint.x"
    - …
  - Animation = [CABasicAnimation animationWithKeyPath:@"..."];
- Add to layer
  - [layer addAnimation:animation];
- Note: Model value is not changed

Explicit Animations

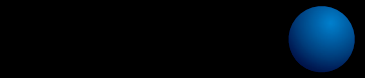
Basic animations

```
// Let’s drop the ball!
CABasicAnimation* move
                = [CABasicAnimation
animationWithKeyPath:@"position.y"];
// Set animation properties
move.duration = 2;
move.toValue = [NSNumber numberWithFloat:300];
[ball addAnimation:move forKey:@"bowl"];
```

Explicit Animation

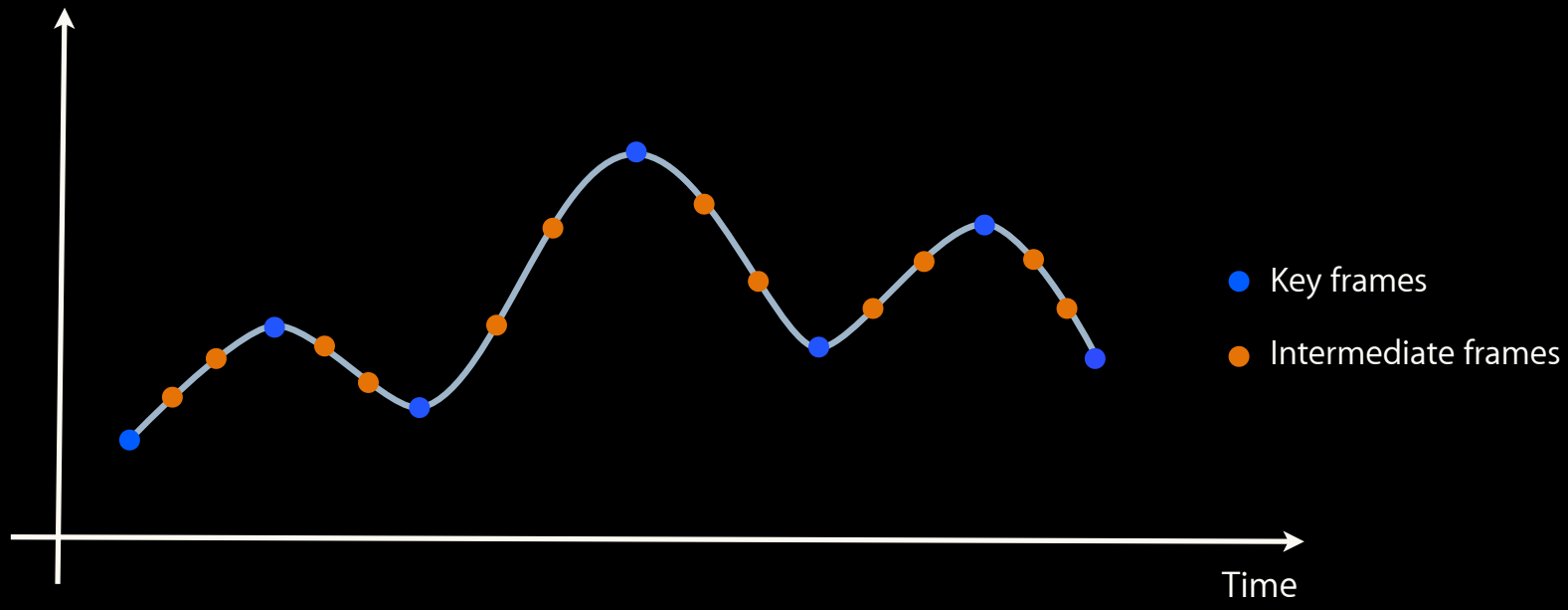
Make it stick

```
// Let’s drop the ball!
CGFloat yAtStart = ball.position.y;
// Target position:
ball.position = CGPointMake(ball.position.x,300);
CABasicAnimation* move = [CABasicAnimation
animationWithKeyPath:@"position.y"];
move.fromValue = [NSNumber numberWithFloat:yAtStart];
move.toValue = [NSNumber numberWithFloat:300];
[ball addAnimation:move forKey:@"position"];
```



Animation

Key frames



Explicit Animations

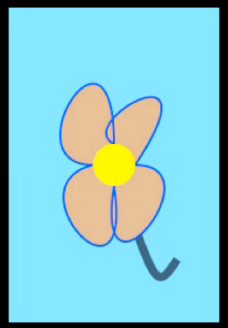
KeyFrame animations

- Use either
  - `path`
  - `values`
- `keyTimes` (optional)—Fraction of total time for each keyframe segment
- Interpolation either between values or along path

Keyframe Animation

```
card.transform = CATransform3DMakeRotation(M_PI, 0, 1, 0);
```

```
CAKeyframeAnimation* animation
     = [CAKeyframeAnimation animationWithKeyPath:@"transform"];
animation.values = [NSArray arrayWithObjects:Zero, Pi, nil];
animation.valueFunction
         = [CAValueFunction functionWithName:kCAValueFunctionRotateY];
animation.duration = 2;
// add it, and override any implicit animation
[card addAnimation:animation @”transform”];
```



Animation Basics

Group animation

- Collection of animations applied simultaneously to a layer's properties
- Timings of animations clipped to groups timing

```
// Move and rotate
CABasicAnimation* move = [CABasicAnimation
       animationWithKeyPath:@"position.x"];
move.toValue = [NSNumber numberWithFloat:x];
move.duration = 10;
CAAnimationGroup* group = [CAAnimationGroup animation];
group.animations = [NSArray arrayWithObjects:move,rotate,nil];
group.duration = 20; // Account for rotation repeats
[card addAnimation:group forKey:nil];
```

Timing

CAMediaTiming protocol

- Adopted by CAAnimation and CALayer
- Properties
  - `beginTime`, `timeOffset`
  - `repeatCount`, `repeatDuration`
  - `duration`, `speed`
  - `autoreverses`
  - `fillMode`

Animation in Real Time

- Remember: Layer model is not changed
- Use presentationLayer to get visible property values

```
CALayer* current = [layer presentationLayer];
if ([current hitTest:touchpoint]) {
     CABasicAnimation* animation = [CABasicAnimation animationWithKeyPath:@"position"];
     animation.fromValue = [NSValue valueWithCGPoint: current.position];
     animation.toValue = [NSValue valueWithCGPoint:someOtherPoint];
     [layer addAnimation: animation forKey:@"position"];
  }
```

Animation Notifications

- Delegate
  - `animationDidStart:`
  - `animationDidStop:finished:`
- Completion block
  - `[CATransaction setCompletionBlock:]`
  - Blocks new in iOS 4
- Use completion to clean up
  - Example: remove layer from parent

Demo

Animation 101 sample

CAAnimation and CALayers

Some more animations and layers

- CATransition
  - Typical use when one layer is replaced by another
  - Animation is applied to parent layer
  - Type: Fade, reveal,moveIn, push
  - SubType: From left, right, top, bottom
- CATextLayer
- CATiledLayer
- CAShapeLayer
- CAReplicatorLayer

Demos

Tim Oriol

Summary

- Layers and their properties
  - Declarative style
  - Properties are animatable
  - Model is 2 1/2 D
- Providing layer content
  - Core Graphics
  - UIKit
  - OpenGL ES
  - AVPlayerLayer
- Animating properties
  - Implicit
  - Explicit

More Information

Allan Schaffer
Graphics and Imaging Evangelist
aschaffer@apple.com

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

Sample Code
https://developer.apple.com/wwdc/iphone/library/samplecode/AnimatedSkeleton/Introduction/Intro.html
http://developer.apple.com/wwdc/iphone/library/samplecode/Animation101/Introduction/Intro.html

Related Sessions

Core Animation in Practice, Part 2	Nob Hill Thursday 2:00PM
Building Animation-Driven Interfaces	Pacific Heights Thursday 9:00AM

Labs

Core Animation Lab	Graphics and Media Lab D Thursday 3:15PM
Animation Lab	Application Frameworks Lab C Thursday 4:30PM


WWDC10