Core Animation

Bonus Lecture 1
Layers

• Core Animation performs drawing using layers (instances/subclasses of CALayer)

• Layers are similar to views
  • A single parent layer
  • (Possibly) many sublayers
  • Drawn back to front
  • Specify geometry relative to superlayer
Layers

- Layers allow more complex geometry
- Rotation, skewing, scaling, projection
- Views contain a layer!
- In fact, layers must be in a view to be displayed
Layer Geometry

- Unlike views, the frame is implicit (computed from the position, bounds, anchorPoint, and transform properties)

- The anchorPoint is expressed in unit coordinates
Layer Transformations

- The transform property (of type CATransform3D) is a 4x4 matrix that is used to apply a variety of transformations to the layer.
There's a lot more!
Animating Layers

• Certain layer properties are designated as animatable (position, opacity, transform, and others)

• Changing the value of a property causes the change to be animated
Implicit Animation

// assume the layer is currently
// positioned at (100, 100)
theLayer.position = CGPointMake(500, 500);
theLayer.opacity = 0;

anotherLayer.opacity = 1;
Explicit Animation

```swift
CABasicAnimation *theAnimation;
theAnimation = [CABasicAnimation animationWithKeyPath: @"opacity"];
theAnimation.duration = 3;
theAnimation.repeatCount = 2;
theAnimation.autoreverses = YES;
theAnimation.fromValue = [NSNumber numberWithInt: 1];
theAnimation.toValue = [NSNumber numberWithInt: 0];
[theLayer addAnimation: theAnimation
   forKey: @"animateOpacity" ];
```
View Animation

• This is still rather a lot of code

• Views, like layers, have certain animatable properties

• Changes to these properties can be animated with only a few extra lines of code
View Animation

Class methods on UIView:

+ beginAnimations:context:
+ commitAnimations
+ setAnimationStartDate:
+ setAnimationsEnabled:
+ setAnimationDelegate:
+ setAnimationWillStartSelector:
+ setAnimationDidStopSelector:
+ setAnimationDuration:
+ setAnimationDelay:
+ setAnimationCurve:
+ setAnimationRepeatCount:
+ setAnimationRepeatAutoreverses:
+ setAnimationBeginsFromCurrentState:
+ setAnimationTransition:forView:cache:
+ areAnimationsEnabled
Demo!
Objective-C Blocks
Objective-C Blocks

- GCC and Clang (since OS X 10.6 and iOS 4) added C-level support for a runtime feature called blocks

- Like functions, but also encapsulate variable bindings to stack and heap
  - Basically full closures

- Particularly useful as callbacks
Example Block

```c
int multiplier = 7;
int (^myBlock)(int) = ^(int num)
{
    return num * multiplier;
}

// example call
printf("%d", myBlock(3)); // prints "21"
```
Example Block

Return type

```c
int multiplier = 7;
int (^myBlock)(int) = ^(int num)
{
    return num * multiplier;
}

// example call
printf("%d", myBlock(3)); // prints "21"
```

These slides are licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.
Example Block

Name of block variable

```c
int multiplier = 7;
int (^myBlock)(int) = ^(int num)
{
    return num * multiplier;
}
```

// example call
printf("%d", myBlock(3)); // prints "21"
Example Block

Parameter type(s)

```c
int multiplier = 7;
int (^myBlock)(int) = ^(int num)
{
    return num * multiplier;
}

// example call
printf("%d", myBlock(3)); // prints "21"
```
Example Block

```
int multiplier = 7;
int (^myBlock)(int) = ^(int num)
{
    return num * multiplier;
}

// example call
printf("%d", myBlock(3)); // prints "21"
```

Parameter names
Inline Blocks

```c

qsort_b(myCharacters, 3, sizeof(char *), ^(const void *l, const void *r)
{
    char *left = *(char **)l;
    char *right = *(char **)r;
    return strncmp(left, right, 1);
});

// myCharacters is now { "Charles Condomine", "George", "TomJohn" }
```
Even Better Animations

• With the addition of Objective-C blocks, Apple introduced new `UIView` methods for performing animations:

  + animateWithDuration:delay:options:animations:completion:
  + animateWithDuration:animations:completion:
  + animateWithDuration:animations:
  + transitionWithView:duration:options:animations:completion:
  + transitionFromView:toView:duration:options:completion:
Animation with Blocks

+ (void) animateWithDuration: (NSTimeInterval) duration
delay: (NSTimeInterval) delay
options: (UIViewAnimationOptions) options
animations: (void (^)(void)) animations
completion: (void (^)(BOOL finished)) completion
UIViewAnimationOptions

```
enum {
    UIViewAnimationOptionLayoutSubviews = 1 << 0,
    UIViewAnimationOptionAllowUserInteraction = 1 << 1,
    UIViewAnimationOptionBeginFromCurrentState = 1 << 2,
    UIViewAnimationOptionRepeat = 1 << 3,
    UIViewAnimationOptionAutoreverse = 1 << 4,
    UIViewAnimationOptionOverrideInheritedDuration = 1 << 5,
    UIViewAnimationOptionOverrideInheritedCurve = 1 << 6,
    UIViewAnimationOptionAllowAnimatedContent = 1 << 7,
    UIViewAnimationOptionShowHideTransitionViews = 1 << 8,
    UIViewAnimationOptionCurveEaseInOut = 0 << 16,
    UIViewAnimationOptionCurveEaseIn = 1 << 16,
    UIViewAnimationOptionCurveEaseOut = 2 << 16,
    UIViewAnimationOptionCurveLinear = 3 << 16,
    UIViewAnimationOptionTransitionNone = 0 << 20,
    UIViewAnimationOptionTransitionFlipFromLeft = 1 << 20,
    UIViewAnimationOptionTransitionFlipFromRight = 2 << 20,
    UIViewAnimationOptionTransitionCurlUp = 3 << 20,
    UIViewAnimationOptionTransitionCurlDown = 4 << 20,
    UIViewAnimationOptionTransitionCrossDissolve = 5 << 20,
    UIViewAnimationOptionTransitionFlipFromTop = 6 << 20,
    UIViewAnimationOptionTransitionFlipFromBottom = 7 << 20,
};
typedef NSUInteger UIViewAnimationOptions;
```
Demo!
Resources

- Core Animation Programming Guide

- Blocks Programming Topics

- UIView Class Reference