Animations

In iOS, creating sophisticated animation does not require you to write any drawing code.

What Can Be Animated?
1. Both UIKit and Core Animation provide support for animations. In UIKit, animations performed using UIView objects.

2. Animatable UIView properties

Property	Changes you can make
`frame`	Modify this property to change the view's size and position relative to its superview's coordinate system. (If the `transform` property does not contain the identity transform, modify the `bounds` or `center` properties instead.)
`bounds`	Modify this property to change the view's size.
`center`	Modify this property to change the view's position relative to its superview's coordinate system.
`transform`	Modify this property to scale, rotate, or translate the view relative to its center point. Transformations using this property are always performed in 2D space. (To perform 3D transformations, you must animate the view's layer object using Core Animation.)
`alpha`	Modify this property to gradually change the transparency of the view.
`backgroundColor`	Modify this property to change the view's background color.

`contentStretch`	Modify this property to change the way the view's contents are stretched to fill the available space.

3. Using Core Animation, you can animate the following types of changes for your view's layer:
=> The size and position of the layer
=> The center point used when performing transformations
=> Transformations to the layer or its sublayers in 3D space
=> The addition or removal of a layer from the layer hierarchy
=> The layer's Z-order relative to other sibling layers
=> The layer's shadow
=> The layer's border (including whether the layer's corners are rounded)
=> The portion of the layer that stretches during resizing operations
=> The layer's opacity
=> The clipping behavior for sublayers that lie outside the layer's bounds
=> The current contents of the layer
=> The rasterization behavior of the layer

Animating Property Changes in a View
In order to animate changes to a property of the UIView class, you must wrap those changes inside an animation block.

> Starting Animations Using the Block-Based Methods
1. The followings are block-based method that offer different level of configuration for the animation block:
=> animateWithDuration:animations:
=> animateWithDuration:animations:completion:
=> animateWithDuration:delay:options:animations:completion:

2. Performing a simple block-based animation

```
[UIView animateWithDuration:1.0 animations:^{
        firstView.alpha = 0.0;
        secondView.alpha = 1.0;
}];
```

3. Using a completion handler is the primary way that you link multiple animations.

> Starting Animations Using the Begin/Commit Methods

> Configuring the Parameters for Begin/Commit Animations

> Configuring an Animation Delegate

> Nesting Animation Blocks
1. You can assign different timing and configuration options to parts of animation block by nesting additional animation blocks.

2. Nesting animations that have different configurations

```
[UIView animateWithDuration:1.0
      delay: 1.0
      options:UIViewAnimationOptionCurveEaseOut
      animations:^{
         aView.alpha = 0.0;

         // Create a nested animation that has a different
         // duration, timing curve, and configuration.
         [UIView animateWithDuration:0.2
             delay:0.0
             options: UIViewAnimationOptionOverrideInheritedCurve |
                    UIViewAnimationOptionCurveLinear |
                    UIViewAnimationOptionOverrideInheritedDuration |
                    UIViewAnimationOptionRepeat |
                    UIViewAnimationOptionAutoreverse
             animations:^{
                [UIView setAnimationRepeatCount:2.5];
                anotherView.alpha = 0.0;
             }
             completion:nil];

      }
      completion:nil];
```

> Implementing Animations That Reverse Themselves
When creating reversible animations in conduction with a repeat count, consider a non integer value for the repeat count.

Creating Animated Transitions Between Views
You use view transition to implement the following types of changes:

=> Change the visible subviews of an existing view.
=> Replace one view in your view hierarchy with a different view.

> Changing the Subviews of a View
1. you use the
transitionWithView:duration:options:animations:completion: method to initiate a transition animation for a view.

2. Listing below is an example of how to use a transition animation to make it seem as if a new text entry page has been added.

```
- (IBAction)displayNewPage:(id)sender
{
    [UIView transitionWithView:self.view
        duration:1.0
        options:UIViewAnimationOptionTransitionCurlUp
        animations:^{
            currentTextView.hidden = YES;
            swapTextView.hidden = NO;
        }
        completion:^(BOOL finished){
            // Save the old text and then swap the views.
            [self saveNotes:temp];

            UIView*    temp = currentTextView;
            currentTextView = swapTextView;
            swapTextView = temp;
        }];
}
```

> Replacing a View with a Different View
1. Replacing a view is something you do when you want your interface to be different.

2. In iOS 4 and later, you use the
transitionFromView:toView:duration:options:completion: method to transition between two views.

3. Toggling between two views in a view controller

```
- (IBAction)toggleMainViews:(id)sender {
    [UIView transitionFromView:(displayingPrimary ? primaryView :
secondaryView)
```

```
        toView:(displayingPrimary ? secondaryView : primaryView)
        duration:1.0
        options:(displayingPrimary ?
UIViewAnimationOptionTransitionFlipFromRight :
                    UIViewAnimationOptionTransitionFlipFromLeft)
        completion:^(BOOL finished) {
            if (finished) {
                displayingPrimary = !displayingPrimary;
            }
    }];
}
```

Linking Multiple Animations Together
For block-based animations, use the completion handler supported by the animateWithDuration:animations:completion: and animateWithDuration:delay:options:animations:completion: methods to execute any follow-on animations.

Animating View and Layer Changes Together
1. Application can freely mix view-based and layer-based animation code as needed but the process for configuring your animation parameters depends on who owns the layer.

2. If you want to customize the animation parameters for layers you create, you must use Core Animation directly.

3. Mixing view and layer animations

```
[UIView animateWithDuration:1.0
    delay:0.0
    options: UIViewAnimationOptionCurveLinear
    animations:^{
        // Animate the first half of the view rotation.
        CGAffineTransform  xform =
CGAffineTransformMakeRotation(DEGREES_TO_RADIANS(-180));
        backingView.transform = xform;

        // Rotate the embedded CALayer in the opposite direction.
        CABasicAnimation*    layerAnimation = [CABasicAnimation
animationWithKeyPath:@"transform"];
        layerAnimation.duration = 2.0;
        layerAnimation.beginTime = 0; //CACurrentMediaTime() + 1;
```

```
        layerAnimation.valueFunction = [CAValueFunction
functionWithName:kCAValueFunctionRotateZ];
        layerAnimation.timingFunction = [CAMediaTimingFunction
                        functionWithName:kCAMediaTimingFunctionLinear];
        layerAnimation.fromValue = [NSNumber numberWithFloat:0.0];
        layerAnimation.toValue = [NSNumber
numberWithFloat:DEGREES_TO_RADIANS(360.0)];
        layerAnimation.byValue = [NSNumber
numberWithFloat:DEGREES_TO_RADIANS(180.0)];
        [manLayer addAnimation:layerAnimation
forKey:@"layerAnimation"];
    }
    completion:^(BOOL finished){
        // Now do the second half of the view rotation.
        [UIView animateWithDuration:1.0
            delay: 0.0
            options: UIViewAnimationOptionCurveLinear
            animations:^{
                CGAffineTransform  xform =
CGAffineTransformMakeRotation(DEGREES_TO_RADIANS(-359));
                backingView.transform = xform;
            }
            completion:^(BOOL finished){
                backingView.transform = CGAffineTransformIdentity;
         }];
}];
```