Key-Value Coding Extensions

Core Animation extends the `NSKeyValueCoding` protocol as it pertains to the `CAAnimation` and `CALayer` classes. This extension adds default values for some keys, expands wrapping conventions, and adds key path support for `CGPoint`, `CGRect`, `CGSize`, and `CATransform3D` types.

Key-Value Coding Compliant Container Classes

The `CAAnimation` and `CALayer` classes are key-value coding compliant container classes, which means that you can set values for arbitrary keys. Even if the key `someKey` is not a declared property of the `CALayer` class, you can still set a value for it as follows:

```
[theLayer setValue:[NSNumber numberWithInteger:50] forKey:@"someKey"];
```

You can also retrieve the value for arbitrary keys like you would retrieve the value for other key paths. For example, to retrieve the value of the `someKey` path set previously, you would use the following code:

```
someKeyValue=[theLayer valueForKey:@"someKey"];
```

OS X Note: The `CAAnimation` and `CALayer` classes, which automatically archive any additional keys that you set up for instances of those classes, support the `NSCoding` protocol.

Default Value Support

Core Animation adds a convention to key value coding whereby a class can provide a default value for a key that has no set value. The `CAAnimation` and `CALayer` classes support this convention using the `defaultValueForKey:` class method.

To provide a default value for a key, create a subclass of the desired class and override its `defaultValueForKey:` method. Your implementation of this method should examine the key parameter and return the appropriate default value. Listing C-1 shows a sample implementation of the `defaultValueForKey:` method for a layer object that provides a default value for the `masksToBounds` property.

Listing C-1 Example implementation of defaultValueForKey:

```
+ (id)defaultValueForKey:(NSString *)key
{
    if ([key isEqualToString:@"masksToBounds"])
         return [NSNumber numberWithBool:YES];

    return [super defaultValueForKey:key];
}
```

Wrapping Conventions

When the data for a key consists of a scalar value or C data structure, you must wrap that type in an object before assigning it to the layer. Similarly, when accessing that type, you must retrieve an object and then unwrap the appropriate values using the extensions to the appropriate class. Table C-1 lists the C types commonly used and the Objective-C class you use to wrap them.

Table C-1 Wrapper classes for C types

C type	Wrapping class
`CGPoint`	`NSValue`
`CGSize`	`NSValue`
`CGRect`	`NSValue`
`CATransform3D`	`NSValue`
`CGAffineTransform`	`NSAffineTransform` (OS X only)

Key Path Support for Structures

The `CAAnimation` and `CALayer` classes lets you access the fields of selected data structures using key paths. This feature is a convenient way to specify the field of a data structure that you want to animate. You can also use these conventions in conjunction with the `setValue:forKeyPath:` and `valueForKeyPath:` methods to set and get those fields.

CATransform3D Key Paths

You can use the enhanced key path support to retrieve specific transformation values for a property that contains a `CATransform3D` data type. To specify the full key path for a layer's transforms, you would use the string value `transform` or `sublayerTransform` followed by one of the field key paths in Table C-2. For example, to specify a rotation factor around the layer's z axis, you would specify the key path `transform.rotation.z`.

Table C-2 Transform field value key paths

Field Key Path	Description
`rotation.x`	Set to an `NSNumber` object whose value is the rotation, in radians, in the x axis.
`rotation.y`	Set to an `NSNumber` object whose value is the rotation, in radians, in the y axis.
`rotation.z`	Set to an `NSNumber` object whose value is the rotation, in radians, in the z axis.
`rotation`	Set to an `NSNumber` object whose value is the rotation, in radians, in the z axis. This field is identical to setting the `rotation.z` field.
`scale.x`	Set to an `NSNumber` object whose value is the scale factor for the x axis.
`scale.y`	Set to an `NSNumber` object whose value is the scale factor for the y axis.
`scale.z`	Set to an `NSNumber` object whose value is the scale factor for the z axis.

`scale`	Set to an `NSNumber` object whose value is the average of all three scale factors.
`translation.x`	Set to an `NSNumber` object whose value is the translation factor along the x axis.
`translation.y`	Set to an `NSNumber` object whose value is the translation factor along the y axis.
`translation.z`	Set to an `NSNumber` object whose value is the translation factor along the z axis.
translation	Set to an `NSValue` object containing an `NSSize` or `CGSize` data type. That data type indicates the amount to translate in the x and y axis.

The following example shows how you can modify a layer using the `setValue:forKeyPath:` method. The example sets the translation factor for the x axis to 10 points, causing the layer to shift by that amount along the indicated axis.

```
[myLayer setValue:[NSNumber numberWithFloat:10.0]
forKeyPath:@"transform.translation.x"];
```

Note: Setting values using key paths is not the same as setting them using Objective-C properties. You cannot use property notation to set transform values. You must use the `setValue:forKeyPath:` method with the preceding key path strings.

CGPoint Key Paths

If the value of a given property is a `CGPoint` data type, you can append one of the field names in Table C-3 to the property to get or set that value. For example, to change the x component of a layer's `position` property, you could write to the key path `position.x`.

Table C-3 CGPoint data structure fields

Structure Field	Description
`x`	The x component of the point.
`y`	The y component of the point.

CGSize Key Paths

If the value of a given property is a `CGSize` data type, you can append one of the field names in Table C-4 to the property to get or set that value.

Table C-4 CGSize data structure fields

Structure Field	Description
`width`	The width component of the size.
`height`	The height component of the size.

CGRect Key Paths

If the value of a given property is a `CGRect` data type, you can append the following field names in Table C-3 to the property to get or set that value. For example, to change the width component of a layer's `bounds` property, you could write to the key path `bounds.size.width`.

Table C-5 CGRect data structure fields

Structure Field	Description
`origin`	The origin of the rectangle as a `CGPoint`.
`origin.x`	The x component of the rectangle origin.
`origin.y`	The y component of the rectangle origin.
`size`	The size of the rectangle as a `CGSize`.
`size.width`	The width component of the rectangle size.
`size.height`	The height component of the rectangle size.