アニメーションのタイプとタイミングのプログラミングガイド

Cocoa > Graphics & Imaging

2010-05-18

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

アップルジャパン株式会社
〒163-1450 東京都新宿区西新宿
3 丁目20 番2 号
東京オペラシティタワー
http://www.apple.com/jp/

目次

図、表、リスト

アニメーションのタイプとタイミングのプログラミングガイドの紹介

この文書では、Core Animationで使用するタイミングとアニメーションクラスに関する基本概念を説明します。Core Animationは、高性能の合成エンジンと使いやすいアニメーションプログラミングインターフェイスを組み合わせた、Objective-Cのフレームワークです。

注： アニメーションは本質的にビジュアルメディアです。HTML版の『*アニメーションのタイプとタイミングのプログラミングガイド*』には、静止画像に加えてQuickTimeムービーが含まれており、アニメーションの例を表示して概念を分かりやすく示しています。PDF版に含まれているのは静止画像のみです。

この文書を読めば、CocoaアプリケーションでのCore Animationの使いかたを理解できます。Core AnimationではObjective-Cのプロパティを多用するため、Objective-C 2.0プログラミング言語を理解していることが前提となります。また、『Key-Value Coding Programming Guide』で説明されているキー値コーディングの知識も必要です。必須ではありませんが、『Quartz 2D Programming Guide』に説明されているQuartz 2Dイメージングテクノロジーの知識も役立ちます。

この書類の構成

『*アニメーションのタイプとタイミングのプログラミングガイド*』の構成は以下のとおりです。

- 「アニメーションクラスロードマップ」（9 ページ）では、アニメーションクラスとタイミングプロトコルを概説します。
- 「タイミング、時空間、およびCAAnimation」（11 ページ）では、Core Animationのタイミングモデルと`CAAnimation`抽象クラスについて詳しく説明します。
- 「プロパティベースのアニメーション」（15 ページ）では、プロパティベースのアニメーション（`CABasicAnimation`および`CAKeyframeAnimation`）について説明します。.
- 「トランジションアニメーション」（21 ページ）では、トランジションアニメーションクラスの`CATransition`について説明します。

関連項目

以下のプログラミングガイドでは、Core Animationによって使われるいくつかのテクノロジーを解説しています。

- 『*Animation Overview*』では、Mac OS Xで使用可能なアニメーションのテクノロジーについて説明しています。
- 『*Core Animation Programming Guide*』では、Core Animationを使った共通の作業について解説したコードを取り上げています。

- 『*Animation Programming Guide for Cocoa*』では、Cocoaアプリケーションに使用できるアニメーション機能を説明しています。

アニメーションクラスロードマップ

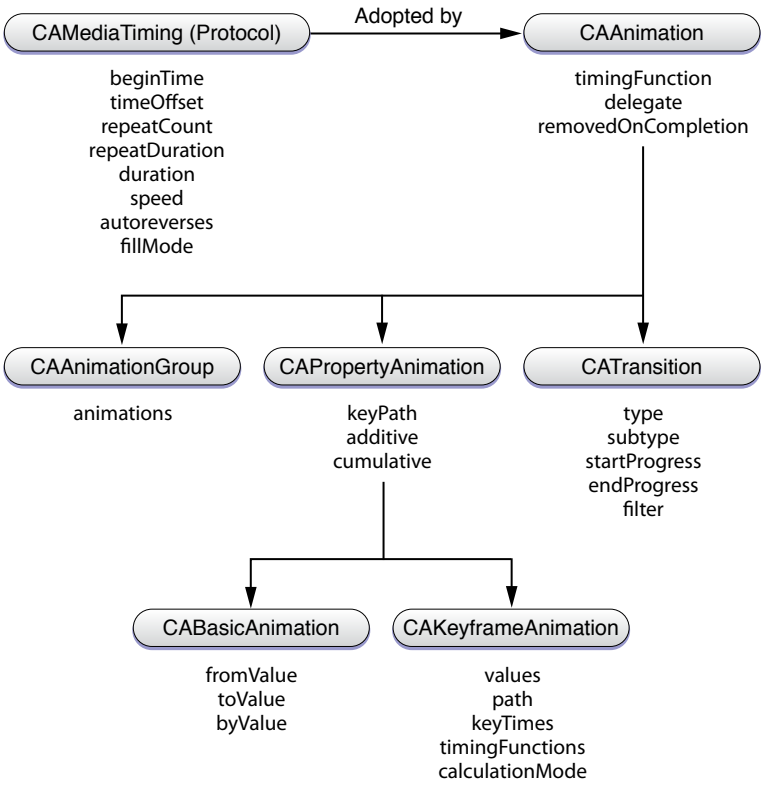
Core Animationでは、アプリケーションで使用できる数々のアニメーションクラスが用意されています。

- CAAnimationは、アニメーションのあらゆるサブクラスの抽象クラスです。CAAnimationは、アニメーションの単純な再生時間、速度、繰り返し回数を指定するCAMediaTimingプロトコルを採用しています。CAAnimationはまた、CAActionプロトコルも採用しています。このプロトコルは、レイヤによってトリガされたアクションに応答して、アニメーションを開始する標準手段を提供します。

  CAAnimationクラスは、CAMediaTimingFunctionのインスタンスとしてアニメーションのタイミングも定義します。タイミング関数は、アニメーションのペーシングを単純なベジエ曲線として記述します。線形のタイミング関数は、アニメーションのペースがその再生時間全体にわたり均一になるように指定し、イーズインのタイミング関数は、アニメーションが再生時間の終わりに近づくにつれて速度を上げるように指定します。
- CAPropertyAnimationは、キーパスによって指定されたレイヤプロパティのアニメーション化を可能にするCAAnimationの抽象サブクラスです。
- CABasicAnimationは、レイヤの単純補間を提供するCAPropertyAnimationのサブクラスです。
- CAKeyframeAnimation（CAPropertyAnimationのサブクラス）は、キーフレームアニメーションをサポートします。アニメーション化するレイヤプロパティのキーパスと、アニメーションの各ステージの値を表す値の配列、およびキーフレームの時間とタイミング関数の配列を指定します。アニメーションが実行するたびに、指定された補間を使って各値が設定されます。
- CATransitionは、レイヤのコンテンツ全体に作用するトランジションエフェクトを提供します。これは、アニメーション化の際にレイヤコンテンツのフェード、プッシュ、リビールを行います。Mac OS Xでは、独自のカスタムCore Imageフィルタを指定することによって、手持ちのトランジションエフェクトを拡張できます。
- CAAnimationGroupを使うと、アニメーションオブジェクトの配列をグループ化して、同時に実行できます。

図 1に、アニメーションクラスの階層を示し、継承を通じて使用できるプロパティの概要も示します。

図 1 Core Animationクラスとプロトコル



タイミング、時空間、CAAnimation

最も単純な定義にまで分解すると、アニメーションとは時間の経過に伴う値の変動にすぎません。Core Animationは、強力なタイムライン機能を提供することで、アニメーションとレイヤの両方に対して基本的なタイミング機能を提供します。

この章では、すべてのアニメーションサブクラスに共通のタイミングプロトコルと基本的なアニメーションサポートについて説明します。

メディアタイミングプロトコル

Core Animationタイミングモデルは`CAMediaTiming`プロトコルによって記述され、`CAAnimation`クラスとそのサブクラスに採用されています。このタイミングモデルでは、アニメーションの時間オフセット、再生時間、速度、および繰り返しを指定します。

`CAMediaTiming`プロトコルは`CALayer`クラスでも採用されており、レイヤは相対的な座標空間を記述するのと同様の方法で、そのスーパーレイヤを基準にした相対的な時空間を定義できます。このレイヤツリーによる時空間の概念が、ルートレイヤからその子孫に渡り、スケーラブルなタイムラインをもたらします。アニメーションは表示先となるレイヤに関連付けられている必要があるため、そのレイヤで定義されている時空間に合わせてアニメーションのタイミングが拡大縮小されます。

アニメーションやレイヤの`speed`プロパティによって、この拡大縮小率を指定します。たとえば、速度値(speed)が2の時空間を持つレイヤにアタッチされた10秒のアニメーションの場合、表示は5秒になります（2倍速）。このレイヤのサブレイヤでも速度係数が2と定義されている場合、そのアニメーションは4分の1の時間で表示されます（スーパーレイヤの速度×サブレイヤの速度）。

同様に、レイヤの時空間はQuartz Composerコンポジションのような動的レイヤメディアの再生にも影響する可能性があります。`QCCompositionLayer`の速度を2倍にすると、コンポジションの再生速度が2倍になり、そのレイヤにアタッチされたあらゆるアニメーションの速度も2倍になります。また、この影響は階層的であるため、`QCCompositionLayers`のサブレイヤもすべて速い速度でコンテンツを表示します。

`CAMediaTiming`プロトコルの`duration`プロパティをアニメーションに対して使用して、アニメーションの1回の再生時間を秒単位で定義できます。`CAAnimation`のサブクラスのデフォルトの再生時間は0秒です。これは、対象のアニメーションはそれが実行されているトランザクションによって指定された再生時間を使う必要があり、トランザクションの時間が指定されていない場合は0.25秒を使用する必要があることを示しています。

タイミングプロトコルはアニメーションを開始する手段を提供します。`beginTime`および`timeOffset`という2つのプロパティを使用して、特定の秒数を再生時間に指定します。`beginTime`は、アニメーションを開始する秒数を指定します。この値は、アニメーションのレイヤの時空間に合わせて拡大縮小されます。`timeOffset`は追加のオフセットを指定しますが、アクティブな現地時間で指定されます。この2つの値を組み合わせて、最終的な開始オフセットが決定されます。

アニメーションの繰り返し

アニメーションの反復動作もまた、CAMediaTimingプロトコルのrepeatCountおよびrepeatDurationプロパティで定義されます。repeatCountはアニメーションの反復回数を指定し、小数値の指定が可能です。10秒のアニメーションのrepeatCountに2.5の値を設定すると、アニメーションは合計25秒実行し、3回目の繰り返しの途中で終了します。repeatCountに1e100fを設定すると、アニメーションは、レイヤから削除されるまで繰り返されます。

repeatDurationはrepeatCountと似ていますが、繰り返し回数ではなく秒単位で指定されます。repeatDurationも小数値をとることができます。

アニメーションのautoreversesプロパティは、複数回の繰り返しが指定されている場合に、アニメーションの前方再生の終了後、逆方向に再生するかどうかを指定します。

フィルモード

タイミングプロトコルのfillModeプロパティは、アクティブな再生時間以外のときにアニメーションをどのように表示するかを定義します。アニメーションは、その開始位置、終了位置、またはその両方で停止させたり、表示から完全に削除したりできます。デフォルトの動作では、アニメーションが終了したら表示から削除するようになっています。

アニメーションのペーシング

アニメーションのペーシングは、補間された値がアニメーションの再生時間中にどのように配分されるかを決定します。ある特定の視覚効果に対して適切なペーシングを使用することで、ユーザに対するその効果を大いに高めることができます。

アニメーションのペーシングは、三次元ベジエ曲線で表されるタイミング関数によって表現されます。この関数は、アニメーションの1回の再生時間（[0.0,1.0]の範囲に正規化）を出力時間（同様の範囲に正規化）にマップします。

CAAnimationクラスのtimingFunctionプロパティは、CAMediaTimingFunctionのインスタンスを指定します。このクラスはタイミング機能をカプセル化します。

CAMediaTimingFunctionでは、マップ関数を指定する際の2つの選択肢が用意されています。すなわち、一般的なペーシング曲線を作成する定数と、2つの制御点を指定することによってカスタム関数を作成するメソッドです。

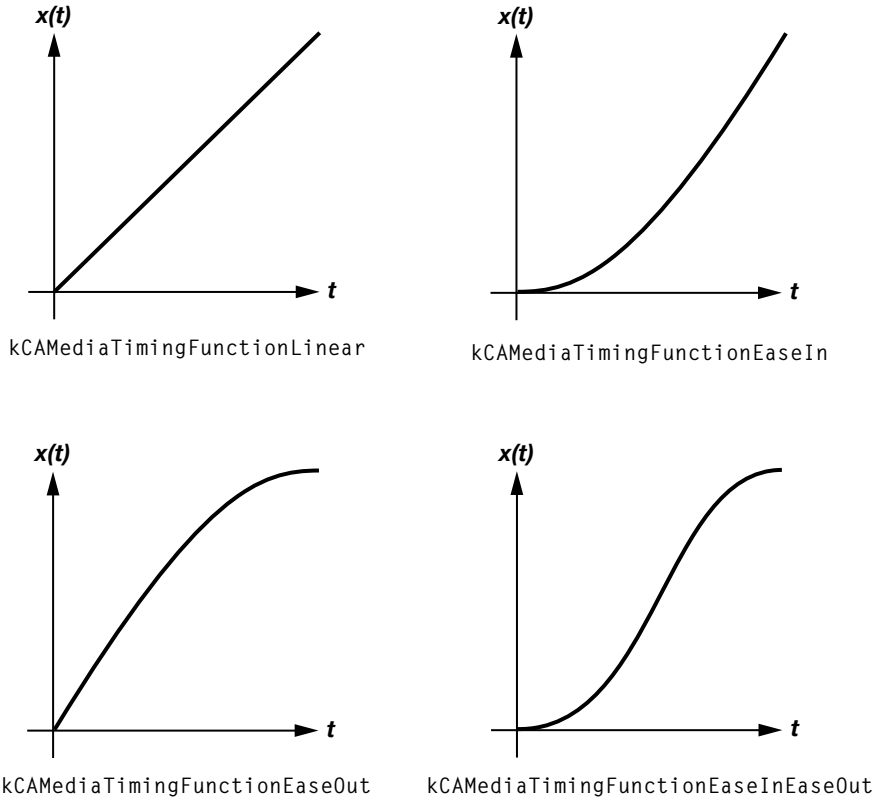
以下の定数の1つをCAMediaTimingFunctionクラスメソッドのfunctionWithName:に指定すると、あらかじめ定義されているタイミング関数が返されます。

- kCAMediaTimingFunctionLinearは、線形のペーシングを指定します。線形のペーシングはアニメーションをその再生時間全体に渡り均等に実行します。
- kCAMediaTimingFunctionEaseInはイーズインのペーシングを指定します。イーズインのペーシングは、アニメーションを低速で開始し、その後加速します。
- kCAMediaTimingFunctionEaseOutは、イーズアウトのペーシングを指定します。イーズアウトのペーシングは、最初は高速で開始し、その後終了に向かって減速します。

- kCAMediaTimingFunctionEaseInEaseOutは、イーズインイーズアウトのペーシングを指定します。イーズインイーズアウトアニメーションは、低速で開始し、再生の途中で加速し、終了前に再度減速します。

図 1に定義済みのタイミング関数を三次元ベジエタイミング曲線で示します。

図 1　定義済みのタイミング関数の三次元ベジエ曲線表現



カスタムのタイミング関数は、functionWithControlPoints::::クラスメソッド、またはinitWithControlPoints::::インスタンスメソッドを使用して作成します。ベジエ曲線の終点は、自動的に(0.0,0.0)および(1.0,1.0)に設定されています。作成メソッドでは、パラメータとしてc1x、c1y、c2x、およびc2yが期待されます。結果として得られる、ベジエ曲線を定義する制御点は[(0.0,0.0), (c1x,c1y), (c2x,c2y), (1.0,1.0)]です。

リスト 1に、点[(0.0,0.0), (0.25,0.1), (0.25,0.1), (1.0,1.0)]を使用してカスタムタイミング関数を作成するコード例を示します。

リスト 1　CAMediaTimingFunctionのカスタムコード

```
CAMediaTimingFunction *customTimingFunction;
customTimingFunction=[CAMediaTimingFunction functionWithControlPoints:0.25f
:0.1f :0.25f :1.0f];
```

注： キーフレームアニメーションでは、`CAMediaTimingFunction`の1つのインスタンスによって提供されるものよりも複雑なペーシングとタイミングのモデルが必要です。詳細については「キーフレームのタイミングとペーシングの拡張」（19 ページ）を参照してください。

アニメーションデリゲート

`CAAnimation`クラスは、アニメーションの開始時および停止時に、デリゲートオブジェクトを通知する手段を提供します。

アニメーションにデリゲートが指定されていると、アニメーションは、開始したアニメーションインスタンスが指定された`animationDidStart:`メッセージを受け取ります。アニメーションが停止するとデリゲートは、停止したアニメーションインスタンスとアニメーションの再生が完全に終了したのか手動で停止されたのか示すBoolean値が指定された`animationDidStop:finished:`メッセージを受け取ります。

重要： `CAAnimation`デリゲートオブジェクトはレシーバ側で保持されます。これは、『*Memory Management Programming Guide*』で説明しているメモリ管理規則のごくまれな例外です。

プロパティベースのアニメーション

プロパティベースのアニメーションは、たとえば位置、背景色、境界矩形など、1つのレイヤの1つの属性値を補間するアニメーションです。

この章では、Core Animationがどのようにプロパティアニメーションを抽象化しているかと、レイヤプロパティの基本的かつ複数のキーフレームアニメーションをサポートするためにCore Animationが提供しているクラスについて説明します。

プロパティベースの抽象化

`CAPropertyAnimation`クラスは、レイヤの特定のプロパティをアニメーション化するための基本機能を提供する`CAAnimation`の抽象サブクラスです。プロパティベースのアニメーションは、以下を含め、数学的に補間できるすべての値の型に対してサポートされます。

- 整数およびdouble値
- `CGRect`、`CGPoint`、`CGSize`、および`CGAffineTransform`構造体
- `CATransform3D`データ構造体
- `CGColor`および`CGImage`リファレンス

あらゆるアニメーションと同様に、プロパティアニメーションはレイヤに関連付けられている必要があります。アニメーション化されるプロパティは、レイヤに関連付けられたキー値コーディングのキーパスを使用して指定します。たとえば、「layerA」の`position`プロパティのX成分をアニメーション化するには、キーパス「position.x」を使用してアニメーションを作成し、「layerA」にそのアニメーションを追加します。

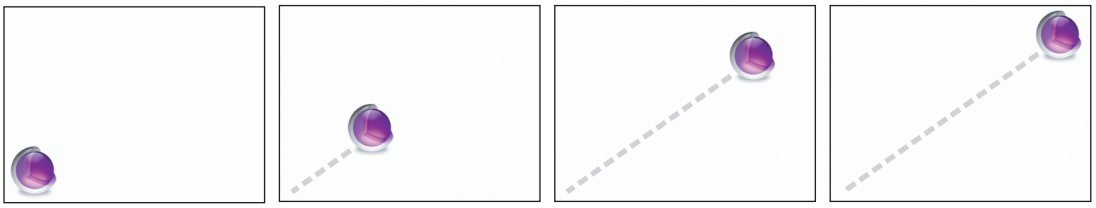
`CAPropertyAnimation`のインスタンスは、直接インスタンス化する必要はありません。代わりに、そのサブクラス、`CABasicAnimation`または`CAKeyframeAnimation`のインスタンスを1つ作成します。さらに、機能の追加や追加プロパティの格納には、`CAPropertyAnimation`をサブクラス化するのではなく、`CABasicAnimation`または`CAKeyframeAnimation`をサブクラス化します。

基本アニメーション

`CABasicAnimation`クラスは、レイヤプロパティに対して基本的な単一キーフレームのアニメーションの機能を提供します。継承した`animationWithKeyPath:`メソッドを使用して、アニメーション化するレイヤプロパティのキーパスを指定することによって`CABasicAnimation`のインスタンスを作成します。『*Core Animation Programming Guide*』の「Animatable Properties」で、`CALayer`のアニメーション化できるプロパティとフィルタプロパティを概説しています。

図 1に、レイヤのpositionプロパティに(74.0,74.0)から(566.0,406.0)の最終位置までを指定した3秒間のアニメーションを示します。親レイヤの矩形領域は(0.0,0.0,640.0,480.0)であるものとします。

図 1　レイヤのpositionプロパティの3秒間基本アニメーション



基本アニメーションの補間値の構成

CABasicAnimationクラスのfromValue、byValue、およびtoValueプロパティは、その間の補間される値を定義します。すべて省略可能で、nil以外の値になるのは2つまでです。プロパティが設定されるオブジェクトタイプは、アニメーション化するプロパティタイプと一致している必要があります。

補間値は、次のように使用されます。

- fromValueとtoValueがnil以外の場合、fromValueとtoValueの間を補間します。
- fromValueとbyValueがnil以外の場合、fromValueと(fromValue+byValue)の間を補間します。
- byValueとtoValueがnil以外の場合、toValueと(byValue-toValue)の間を補間します。
- fromValueがnil以外の場合、fromValueとプロパティの現在のプレゼンテーション値の間を補間します。
- toValueがnil以外の場合、対象レイヤのプレゼンテーションレイヤのkeyPathの現在値とtoValueの間を補間します。
- byValueがnil以外の場合、対象レイヤのプレゼンテーションレイヤのkeyPathの現在値と、その値とbyValueの合計値の間を補間します。
- すべてのプロパティがnilの場合、対象レイヤのプレゼンテーションレイヤの、keyPathの以前の値と、対象レイヤのプレゼンテーションレイヤのkeyPathの現在の値の間を補間します。

注： Mac OS上では、これらのプロパティに渡される値はNSPoint構造体ですが、iOS上ではCGPoints型です。このわずかな違いを除き、使いかたは同じです。

基本アニメーションの例

リスト1に、図1で示したアニメーションと同じ明示的なアニメーションを実装するコードを示します。

リスト 1　CABasicAnimationコード

```
CABasicAnimation *theAnimation;

// positionプロパティをキーパスに指定して、アニメーションオブジェクトを作成する
// キーパスは対象とするアニメーションオブジェクト（この場合はCALayer）を基準にする
theAnimation=[CABasicAnimation animationWithKeyPath:@"position"];

// fromValueおよびtoValueを適切な点に設定する
theAnimation.fromValue=[NSValue valueWithPoint:NSMakePoint(74.0,74.0)];
theAnimation.toValue=[NSValue valueWithPoint:NSMakePoint(566.0,406.0)];

// 再生時間を3.0秒に設定する
theAnimation.duration=3.0;

// カスタムタイミング関数を設定する
theAnimation.timingFunction=[CAMediaTimingFunction functionWithControlPoints:0.25f
 :0.1f :0.25f :1.0f];
```

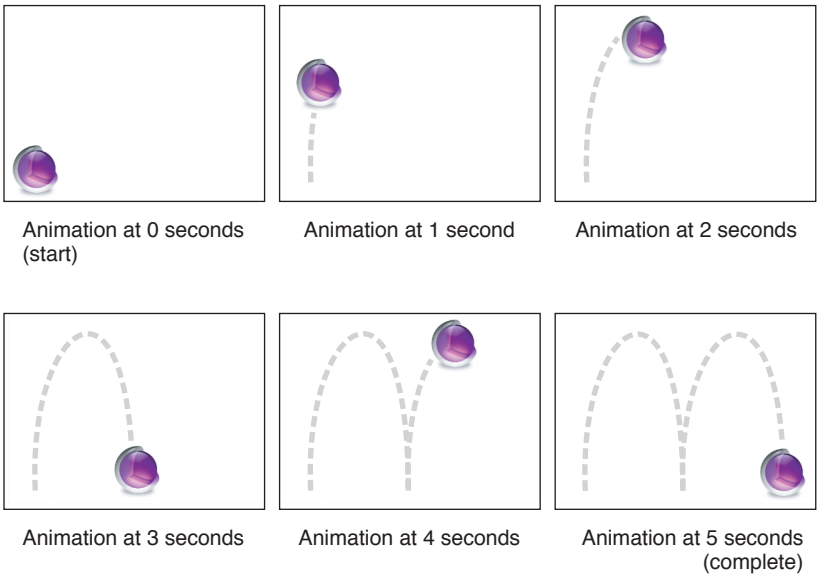
注： この例はMac OS X向けのものです。iOS上でコンパイルするときには、わずかな変更が必要です。NSMakePoint関数は使用できないため、代わりにCGPointMake関数を使います。この直接的な置き換え以外、コードは同じです。

キーフレームアニメーション

Core AnimationのCAKeyframeAnimationクラスでサポートされるキーフレームアニメーションは、基本アニメーションに似ていますが、対象値の配列を指定することができます。これらの対象値は、アニメーションの実行中にそれぞれ順番に補間されます。

図 2に、キーフレーム値にCGPathRefを使ったレイヤのpositionプロパティの5秒間のアニメーションを示します。

図 2 レイヤのpositionプロパティの5秒間キーフレームアニメーション



キーフレーム値の指定

キーフレーム値は、Core Graphicsパス（pathプロパティ）、またはオブジェクトの配列（valuesプロパティ）のどちらかを使って指定します。

Core Graphicsパスは、レイヤのanchorPointまたはpositionプロパティ、つまりCGPointsのプロパティをアニメーション化するのに適しています。moveto点を除くパスの各点は、タイミングと補間のための単一のキーフレームセグメントを定義します。pathが指定されている場合、valuesプロパティは無視されます。

デフォルトでは、レイヤはCGPathに沿ってアニメーション化されるときに、設定された回転を維持します。rotationModeプロパティをkCAAnimationRotateAuto、またはkCAAnimationRotateAutoReverseに設定すると、レイヤはパスの接線と一致するように回転します。

オブジェクトの配列をvaluesプロパティに指定することで、任意のタイプのレイヤプロパティをアニメーション化できます。たとえば、

- CGImageオブジェクトの配列を指定し、アニメーションのキーパスをレイヤのcontentプロパティに設定します。これにより、提供されたイメージ群を通じてレイヤのコンテンツをアニメーション化できます。
- CGRectsの配列（オブジェクトとしてラップされる）を指定し、アニメーションのキーパスをレイヤのframeプロパティに設定します。これにより、渡された矩形を通じてレイヤのフレームを反復させることができます。
- CATransform3D行列の配列（同じく、オブジェクトとしてラップ）を指定、animationキーパスをtransformプロパティに設定します。これにより、変換行列をレイヤのtransformプロパティに順番に適用させることができます。

キーフレームのタイミングとペーシングの拡張

キーフレームアニメーションには、基本的なアニメーションよりも複雑なタイミングとペーシングのモデルが必要です。

継承したtimingFunctionプロパティは無視されます。代わりに、CAMediaTimingFunctionインスタンスの任意指定の配列をtimingFunctionsプロパティに渡すことができます。各タイミング関数は、キーフレームセグメントに対する1つのキーフレームのペーシングを記述します。

継承したdurationプロパティはCAKeyframeAnimationに対して有効ですが、keyTimesプロパティを使うことによってタイミングをより細かく調節できます。

keyTimesプロパティは、各キーフレームセグメントの再生時間を定義するNSNumberオブジェクトの配列を指定します。配列内の各値は0.0から1.0までの浮動小数点数であり、values配列内の1つの要素に対応します。keyTimes配列内の各要素は、対応するキーフレーム値の再生時間を、そのアニメーションの合計再生時間に対する割合として定義します。各要素の値は、直前の値よりも大きいか同じでなければなりません。

keyTimes配列の適切な値はcalculationModeプロパティに依存します。

- calculationModeがkCAAnimationLinearに設定されている場合、配列の先頭の値は0.0で、最後の値は1.0となります。値は指定されたkeytimesの間で補間されます。
- calculationModeがkCAAnimationDiscreteに設定されている場合、配列の最初の値は0.0でなければなりません。
- calculationModeがkCAAnimationPacedに設定されるている場合、keyTimes配列は無視されます。

キーフレームアニメーションの例

リスト2に、図2で示したアニメーションと同じ明示的なアニメーションを実装するコードを示します。

リスト 2　CAKeyframeAnimationコード

```
// 2つの弧（バウンス）を実装するCGPathを作成する
CGMutablePathRef thePath = CGPathCreateMutable();
CGPathMoveToPoint(thePath,NULL,74.0,74.0);
CGPathAddCurveToPoint(thePath,NULL,74.0,500.0,
                                   320.0,500.0,
                                   320.0,74.0);
CGPathAddCurveToPoint(thePath,NULL,320.0,500.0,
                                   566.0,500.0,
                                   566.0,74.0);

CAKeyframeAnimation * theAnimation;

// positionプロパティをキーパスに指定して、アニメーションオブジェクトを作成する
// キーパスは対象とするアニメーションオブジェクト（この場合はCALayer）を基準にする
theAnimation=[CAKeyframeAnimation animationWithKeyPath:@"position"];
theAnimation.path=thePath;

// 再生時間を5.0秒に設定する
theAnimation.duration=5.0;
```

```
// パスを解放する
CFRelease(thePath);
```

トランジションアニメーション

トランジションアニメーションは、レイヤのプロパティ値を変更した影響やレイヤツリー内のレイヤの状態を数学的に補間できない場合に使用します。

この章では、Core Animationが提供するトランジションアニメーション機能について説明します。

トランジションアニメーションの作成

`CATransition`クラスは、Core Animationにトランジション機能を提供します。これは、あるレイヤの特定のプロパティではなく、レイヤ全体に影響する`CAAnimation`の直接のサブクラスです。

`CATransition`の新しいインスタンスは、継承した`animation`クラスメソッドを使用して作成されます。これにより、表1に示すようなデフォルトの値でトランジションアニメーションを作成します。

表 1　`CATransition`のデフォルトプロパティ値

トランジションプロパティ	値
`type`	フェードトランジションを使用。値は`kCATransitionFade`。
`subType`	適用なし。
`duration`	現在のトランザクションのduration、またはトランザクションのdurationが設定されていない場合は0.25秒を使用。値は0.0。
`timingFunction`	線形のペーシングを使用。値は`nil`。
`startProgress`	0.0
`endProgress`	1.0

トランジションの構成

一度作成すると、定義済みのトランジションタイプの1つを使ってトランジションアニメーションを構成したり、Mac OS X上でCore Imageフィルタを使ってカスタムトランジションを作成したりできます。

定義済みのトランジションは、`type`プロパティを表 2に示す定数の1つに設定して使用します。

表 2　一般的なトランジションタイプ

トランジションタイプ	説明
kCATransitionFade	レイヤが表示または非表示になるとフェードする。
kCATransitionMoveIn	レイヤは既存のコンテンツの一番上に徐々に重なっていく。
kCATransitionPush	レイヤが移動するにつれて、既存のコンテンツをすべて押し出す。
kCATransitionReveal	トランジションのサブタイプで指定した方向に、レイヤが徐々に表示される。

kCATransitionFadeを除いて、定義済みのトランジションタイプを使用してトランジションの方向も指定できます。これにはsubTypeプロパティを表 3に示す定数の1つに設定します。

表 3　一般的なトランジションサブタイプ

トランジションサブタイプ定数	説明
kCATransitionFromRight	トランジションはレイヤの右側から始まる。
kCATransitionFromLeft	トランジションはレイヤの左側から始まる。
kCATransitionFromTop	トランジションはレイヤの一番上から始まる。
kCATransitionFromBottom	トランジションはレイヤの一番下から始まる。

startProgressプロパティを使用すると、アニメーション全体のうちの割合を表す値を設定することによってトランジションの開始点を変更できます。たとえば、アニメーションの工程の半分を過ぎたところでトランジションを開始するには、startProgress値を0.5に設定します。同様に、トランジションのendProgress値を指定できます。endProgressは、トランジション全体のうち、トランジションをどの地点で停止するかを示す割合です。デフォルト値は、それぞれ0.0と1.0です。

定義済みのトランジションでは希望する視覚的効果が得られず、アプリケーションがiOSではなくMac OS Xを対象としている場合は、トランジションの表示に使われるカスタムのCore Imageフィルタオブジェクトを指定できます。カスタムフィルタは、入力キーkCIInputImageKeyおよびkCIInputTargetImageKeyと、出力キーkCIOutputImageKeyの両方をサポートしている必要があります。必要に応じて、フィルタはkCIInputExtentKey入力キーをサポートすることもできます。これは、トランジションが実行される領域を示す矩形を設定します。

書類の改訂履歴

この表は「*アニメーションのタイプとタイミングのプログラミングガイド*」の改訂履歴です。

日付	メモ
2010-05-18	サンプルコードにおけるNSMakePointの使いかたについてiOSでの違いについて言及しました。
2010-03-24	ムービーを更新しました。
2008-09-09	誤植を修正しました。
2008-07-08	iOS用に更新しました。
2008-04-08	誤植を修正しました。
2008-02-08	タイミング関数の画像を修正しました。
2007-10-31	Core AnimationのプロキシおよびCocoa Animationのプロキシの両方で使用するアニメーションクラスおよびタイミングクラスについて説明する新規文書。