AnimationKit for REALbasic

Version 2.0 – March 30th, 2008

Introduction

Animated interfaces add some much appreciated glitz to applications. But animation can be a bit difficult, especially when trying to animate based on times rather than frames.

The ZAZ AnimationKit allows developers to animate interface elements – RectControls and Windows (including ContainerControls) using a single line of code. AnimationKit includes convenience methods to retrieve the bounds of an item as a AKRect structure, and create AKTasks which can be triggered for animation later. AnimationKit also allows customizable frame rates, with a default of 32 frames per second.

Simple or complex animation is now easy, thanks to AnimationKit

Usage

AnimationKit includes a module called AKCore, which handles all aspects of the animations. The principal animation class, AKTask, has no real functionality, and must be subclassed to use. AnimationKit contains two such subclasses to handle nearly any type of animation: AKMoveTask and AKFrameTask. Both are similar, but will be covered separately.

AKTask contains a method, NextTask, to allow animation chaining. It can be treated as a property, and read back. When setting NextTask, the new task will be added to the last task in the chain, allowing very easy chaining. There is also an OnCompleteAction property which points to a AKCompletionDelegate to allow code to be executed when an animation completes. Unlike NextTask, OnCompleteAction does not chain. In AKMoveTasks, OnCompleteAction is called when the object reaches it's final position. In AKFrameTasks, OnCompleteAction is fired when the animation completes it's set of frames, and looping is turned off. If looping is on, then OnCompleteAction will not be called. AKTask also has a Cancel method which will stop the animation where it is, no matter what state it is in.

AKMoveTask

AKMoveTask encapsulates all the functionality of the older AnimationTask. It can be created by manually. or by AKCore.NewMoveTask. It can also be created using the Animate extension of Window or RectControl, though Animate will trigger the animation instantly. AKMoveTask's NewRect method describes the final location of the object, using an AKRect. AKRect can be created manually, or by AKCore.CreateRect. Duration is the number of seconds needed to complete the animation and EasingMethod is any of the constants found in AKEasing to describe the animation style – such as Linear and Bounce.

AKFrameTask

Like AKMoveTask, AKFrameTask can be created manually, by AKCore.NewFrameTask, or using the Animate extension of any class which implements the AKFrameTarget interface. Any class can implement AKFrameTarget, but only classes that do can perform frame animations. AKFrameTasks need to be provided with an array of picture objects. The looping property does exactly what it sounds like: if true, the animation will continue over and over again. If false, the animation will complete once each frame has been displayed, and both NextTask and OnCompleteAction will be triggered. AKFrameTasks have a duration property as well, which describe how long it should take to complete one loop.

AKFrameTargets implement a single method, ChangeFrame which provides both the image to display and the frame index to the implementor. There is also an AKAnimatingCanvas which already implements AKFrameTarget to make it very easy to perform frame animations. For example, a custom progress wheel is no more complicated than providing the canvas the set of images, set looping to true, and run the animation. Looping animations should be stopped using the Cancel method if the window is closing, otherwise you are likely to receive NilObjectExceptions.

History

2.0
- All classes have been renamed to a AKxxx naming convention
- Separated AKTask into a "container" class. AKMoveTask replaces AnimationTask.
- Added AKFrameTask, AKFrameTarget, and AKAnimatingCanvas to handle frame-based animations.
- Added AKCompletionDelegate to trigger code when an animation completes.
- Reworked methods to remove need to AKCore.Start and AKCore.Stop methods – this is handled entirely automatically now.

1.2
- Reworked code to use a completely time-based animation method. With the old method, AnimationCore would calculate the number of frames it would take to complete the animation based on the duration and frames per second. This new method guarantees the animation will complete on the duration.

1.1
- Added AnimationEasing module, which provides the constants and methods necessary for animation styles. Special thanks to Stephen Tallent for porting this module to REALbasic.
- Added NextTask property to AnimationTask, allowing chained animation
- Moved the Run method from AnimationTask to AnimationCore. If an AnimationTask for an object is already in the queue, Run will attempt to assign it to NextTask of the existing AnimationTask. This will cause the new animation to run once the current one is finished. If the current task already has a NextTask, Run will not overwrite it, and will then return false.
- Changes to demo project to support new features

1.0.1
- Renamed Rect to AnimationRect to avoid problems caused by namespaces and certain plugins

1.1
- Initial release

Contact

Thom McGrath, thom@thezaz.com
http://www.thezaz.com/

License

2) Simply ask my permission.

If you make changes, feel free to e-mail them to me. If they work out well, I'll roll them into a new release, and give credit where credit is due. But please, don't start distributing modified versions – it creates a real mess.