contact@jara-ai.com

JARA 2D Animation 2.0

JARA 2D Animation v2.0 is a ready to use 2D animation tool for adding animated Sprites to your games with ease.

For how to use the tool on the main project in unity, refer to the file: ***OLD JARA 2D Tutorial***. This tutorial includes the additions of verion 2.0 and how to integrate JARA 2D animation in your code.

NOTE Do not use the methods on page 21 of ***OLD JARA 2D Tutorial***, those are for version 1.0.

1 New Additions

Animated sprites now have two new properties on the Unity *Inspector*: *TerminalPoints* and *LoopStart*.

TerminalPoints When you want to stop your animation, you can do so through code by calling *Stop()* or *StopWithInertia()*. The first one will stop the animation no matter how many frames have elapsed (even if its 1 frame), while the second will wait until the current loop is finished. But what if you don't want to wait for the whole animation to end? *TerminalPoints* allows you to define frames in the animation that can be used as if those where the final frames, stopping the animation when other methods are called. While you don't stop the animation or change it, they have no effect on the behaviour of the animation.

LoopStart What happens if you want your animation to loop, but not from the very start, from a set frame? Well, *LoopStart* lets you define that frame. For example, lets say that your "*Run*" animation has 60 frames, but the first 20 are only for the character preparing to run and the next 40 are the actual run loop. Well, by setting *LoopStart* to the frame 21 (drag and drop your sprite here), your animation will start at frame 1, go until frame 60, and then start another loop from frame 21 to frame 60 (if playing in loop the animation).

2 Playing Animations through code

All of the *Animated Sprites* include a Script that lets you play the animations: **JARA2DAnimationSpriteAnimation**. The following methods can be used to animate an *Animated Sprite*:

Play(string AnimationName, PlaySettings CurrentPlaySettings) This is what you want to call when you want an animation to begin playing. The AnimationName is the name you gave to the animation you want to play when setting up the animated sprite (for example "*Run*"). *CurrentPlaySettings* determines how the animation should be played: *PlaySettings.Once* plays the animation then stops it once finished, *PlaySettings.Loop* will play the animation until the *Stop()* command is called. If *Play()* is called when other animation is playing, the current animation will be stopped and the new one will start from the start.

PlayBlend(string AnimationName, PlaySettings currentPlaySettings) This type of *Play* is used to start playing an animation *AnimationName*. Nevertheless, if another animation is currently running, it will wait until it ends the current cycle to start. If the animation currently playing has *PlaySettings* of *PlaySettings.Loop*, it will end when the current loop ends.

PlaySwap(string AnimationName, PlaySettings currentPlaySettings) This type of *Play* is used to start playing an animation *AnimationName* from the currently playing frame. For example if you are playing "$WalkRight$" and currently is on frame 20 of 60, and want to play "$WalkLeft$", starting

from frame 20 (to make a smooth animation), you can do so by calling this method.

Stop() This is what you want to call to stop the animation that is currently being played. It doesn't matter if the animation is on frame 1 or 50 of 55, the animation will stop.

StopWithInertia() Unlike *Stop*(), this method will wait for the animation to reach the last frame to stop. Useful when you are playing something on a loop but don't want to abruptly stop the animation.

SetDurationInMiliseconds(float NewDurationInMiliseconds) With this, you can manually change the duration of an animation to whatever value you want (in miliseconds).

ChangePlaySpeed(float SpeedFactor) If you just want to accelerate or de-accelerate an animation by an amount, for example double the speed (*ChangePlaySpeed(2)*), or halve it (*ChangePlaySpeed(0.5f)*), you can do so with this function.

ResetPlaySpeed() Undos the changes made by the previous method. It **DOES NOT** revert changes made with *SetDurationInMiliseconds(float NewDurationInMiliseconds)*.

GetCurrentAnimation() Returns the name of the animation currently playing.

GetNextAnimation() Returns the name of the next animation to be played.

To know if an animation is being played at the moment, you can check directly the following variable:

isPlaying