Spritedow Animator

A plugin to do simple sprite animations avoiding the big and tedious Unity's Mecanim system. Oriented to programmers, if you prefer visual scripting you maybe prefer using Mecanim instead of this.

Installation Guide

Simply copy the files on your project or use the UnityPackage included.

Creating an animation

Use the animation editor to create new animation files. You can open it selecting **Sprite Animation Editor** on **Tools/Elendow** tab.

- Give a name to the animation. This will be also the asset name.
- Select the folder to save.
- Select the framerate of the animation.
  - You can also use the animator framerate instead when playing the animation.
- Add frames manually or dropping the sprite to the Drag&Drop box.
  - If you drop a Texture instead of a sprite to the Drag&Drop box, the plugin will take all the sprites on that Texture
  - You can change the duration of each frame, 1 by default, to any number greater than 0.
  - You can sort the frames.
  - You can add actions to the frames, this will be triggered when the animation reaches this frame and you can subscribe to the action to do something (for example, spawning a particle or playing a sound).
- Any change is automatically saved.
- You can preview the animation.
  - The speed and loop settings of the preview window are only for that window.

Inspector properties

- **Ignore time scale** will set the animation to ignore the game TimeScale.
- **Enable render on play** will enable the render each time the animation plays.
- **Disable render on finish** will disable the renderer after every loop cycle if there's a delay specified or will disable the renderer after the animation ends if it's not looping.
- **Play on awake** will start playing when the object awakes.
    - **Start at random frame** will set the animation to start at random frame when "Play" is called.
    - **Backwards** if true the animation will play backwards.
    - **Random Animation** if true the start animation will be random, and the animation will randomly change after every loop cycle.
    - **Animation** if the animation is not random, this animation will play on awake.
    - **One shot** if false the animation will loop infinite times.
        - **Loop Type** repeat will loop the animation resetting it, yoyo will loop going back and forth.
        - **Delay** if true a delay between loops will be made
            - **Min** minimum delay time
            - **Max** maximum delay time
- **Random Animation List** the list of animations used if the animator has play on awake and random animation.
- **Fallback Animation** if a fallback animation is specified, when any animation finish this animation will be played. Useful for idle animations.

UI Image Animator Inspector properties

- **Adapt UI Pivot** with this option selected the pivot of the UI will move using the Sprite pivot. This is useful if you have an animation with multiple canvas sizes but you want it static on one place.

Using the animations

Add the **SpriteAnimator** or **UIAnimator** component to the object you want to animate. This component requires a SpriteRenderer or Image component to work. If the object doesn't have one, the animator will add it automatically. On your code, use **GetComponent<SpriteAnimator>** or **GetComponent<UIAnimator>** to get the reference and start using it.

Animator Methods

- **Play(SpriteAnimation animation, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)**
  - Plays the specified animation.
- **PlayRandom(bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)**
  - Plays a random animation of the random animation list.
- **PlayStartingAtFrame(SpriteAnimation animation, int frame, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)**
  - Plays an animation starting at the specified frame.
- **PlayStartingAtTime(SpriteAnimation animation, float time, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)**
  - Plays an animation starting at the specified time (in seconds).
- **PlayStartingAtNormalizedTime(SpriteAnimation animation, float normalizedTime, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)**
  - Plays an animation starting at the specified normalized time (between 0 and 1).
- **Resume()**
  - Resumes the animation.
- **StopAtFrame()**
  - Stops when reaches the desired frame. If the desired frame has already passed and the animation is not looped it will stop at the end of the animation anyway.
- **Stop()**
  - Stops the animation.
- **SetFallbackAnimation(SpriteAnimation animation, LoopType loopType)**
  - Sets the fallback animation to play and its loop type.
- **UseAnimatorFPS(int frameRate)**
  - Sets the animator FPS overriding the FPS of the animation.

- **UseAnimationFPS()**
  - Sets de animator FPS to the current animation FPS.
- **RemoveFallbackAnimation()**
  - Removes the fallback animation.
- **Restart()**
  - Restarts the animation. If the animation is not playing the effects will apply when starts playing.
- **AddCustomEvent(SpriteAnimation animation, int frame)**
  - Adds a custom event to specified animation on a certain frame.
  - Returns the event created. Null if the animation is null or doesn't have enough frames.
- **AddCustomEventAtEnd(SpriteAnimation animation)**
  - Adds a custom event to specified animation on the last frame.
  - Returns the event created. Null if the animation is null.
- **GetCustomEvent(SpriteAnimation animation, int frame)**
  - Gets the custom event of an animation on a certain frame.
  - Returns the event of the specified animation on the selected frame. Null if not found.
- **GetCustomEventAtEnd(SpriteAnimation animation)**
  - Gets the custom event of an animation on the last frame.
  - Returns the event of the specified animation on the last frame. Null if not found.
- **SetActiveRenderer(bool active)**
  - Enable or disable the renderer.
- **FlipSpriteX(bool flip)**
  - Flip the sprite on the X axis.
  - **Not working on the UIAnimator**
- **FlipSpriteY(bool flip)**
  - Flip the sprite on the Y axis.
  - **Not working on the UIAnimator**
- **SetRandomDelayBetweenLoops(float min, float max)**
  - Sets a random delay between loops.
- **SetDelayBetweenLoops(float delay)**
  - Sets a delay between loops.
- **SetAnimationTime(float time)**
  - Sets the animation time to the specified time, updating de sprite to the correspondent frame at that time.
- **SetAnimationNormalizedTime(float normalizedTime)**
  - Sets the animation time to the specified normalized time (between 0 and 1), updating de sprite to the correspondent frame at that time.

Animator Properties

- **bool IsPlaying { get; }**
  - Gets if the animator is playing an animation or not.
- **bool DisableRenderOnFinish { set; }**
  - If true, the animator will disable the renderer when the animation ends.
- **bool RandomAnimation { set; }**
  - If true the animator will get a random animation after every loop cycle.
- **bool DelayBetweenLoops { set; }**
  - If true a delay will be made between loops.
- **bool IgnoreTimeScale { set; }**
  - If true, the timescale of the game will be ignored.
- **bool StartAtRandomFrame { set; }**
  - If true, the timescale of the game will be ignored.
- **bool StartAtRandomFrame { set; }**
  - The animation will start at a random frame if this is true.
- **int CurrentFrame { get; }**
  - The current frame of the animation.
- **int CurrentFrameRate { get; }**
  - The current FPS of the animator (it could be the animation FPS or an overrided FPS).
- **float CurrentAnimationTime { get; }**
  - The current time in seconds of the playing animation.
- **float CurrentAnimationTimeNormalized { get; }**
  - The current time of the playing animation normalized (between 0 and 1).
- **SpriteAnimation PlayingAnimation { get; }**
  - The currently playing animation.

Animation Actions

- Actions will be called if they are defined on the animation.

```
private SpriteAnimator spriteAnimator;

private void Awake()
{
    spriteAnimator = GetComponent<SpriteAnimator>();
    spriteAnimator.OnAnimationAction += OnAnimactionAction;
}

private void OnAnimactionAction(SpriteAnimationAction action, SpriteAnimation animation)
{
    // Do something
}
```

Animator Events

- Some events will be always triggered.
- **OnFinish** is triggered when the animation reach the last frame.
- **OnPlay** is triggered when the animation starts playing.
- **OnStop** is triggered when the animation is forced to stop.

```
private SpriteAnimator spriteAnimator;

private void Awake()
{
    spriteAnimator = GetComponent<SpriteAnimator>();
    spriteAnimator.OnPlay += OnAnimationPlay;
}

private void OnAnimationPlay()
{
    // Do something
}
```

- You can add custom events to be fired. This will not be saved on the animation.

```
public SpriteAnimation walkAnimation;
private SpriteAnimator spriteAnimator;

private void Awake()
{
    spriteAnimator = GetComponent<SpriteAnimator>();
    spriteAnimator.AddCustomEvent(walkAnimation, 5).AddListener(StepEvent);
}

private void StepEvent(BaseAnimator animator)
{
    // Do something
}
```

Animation Methods

- **Sprite GetFrame (int frame)**
  - Returns the sprite on the selected frame.
- **int GetFrameDuration(int index)**
  - Returns the duration (in frames) of the selected frame.
- **int GetFrameAtTime(float time)**
  - Get the frame at the specified time using the animation frame rate.
- **int GetFrameAtTime(float time, int frameRate)**
  - Get the frame at the specified time using the specified frame rate.
- **float GetAnimationDurationInSeconds()**
  - Get the total duration of the animation in seconds using the animation frame rate.
- **float GetAnimationDurationInSeconds(int frameRate)**
  - Get the total duration of the animation in seconds using the specified frame rate.
- **int GetFrameAtNormalizedTime(float normalizedTime)**
  - Get the frame index at the specified normalized time (between 0 and 1) using the animation frame rate.
- **int GetFrameAtNormalizedTime(float normalizedTime, int frameRate)**
  - Get the frame index at the specified normalized time (between 0 and 1) using the specified frame rate.

Animation Properties

- **int FPS { get; set; }**
  - The framerate of the animation.
- **int FramesCount { get; }**
  - Frames on this animation.
- **List Frames { get; set; }**
  - List of frames.
- **List Actions { get; }**
  - List of custom actions added to the animation.
- **int AnimationDuration { get; }**
  - The total duration of the animation (in frames)