Spritedow Animator

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

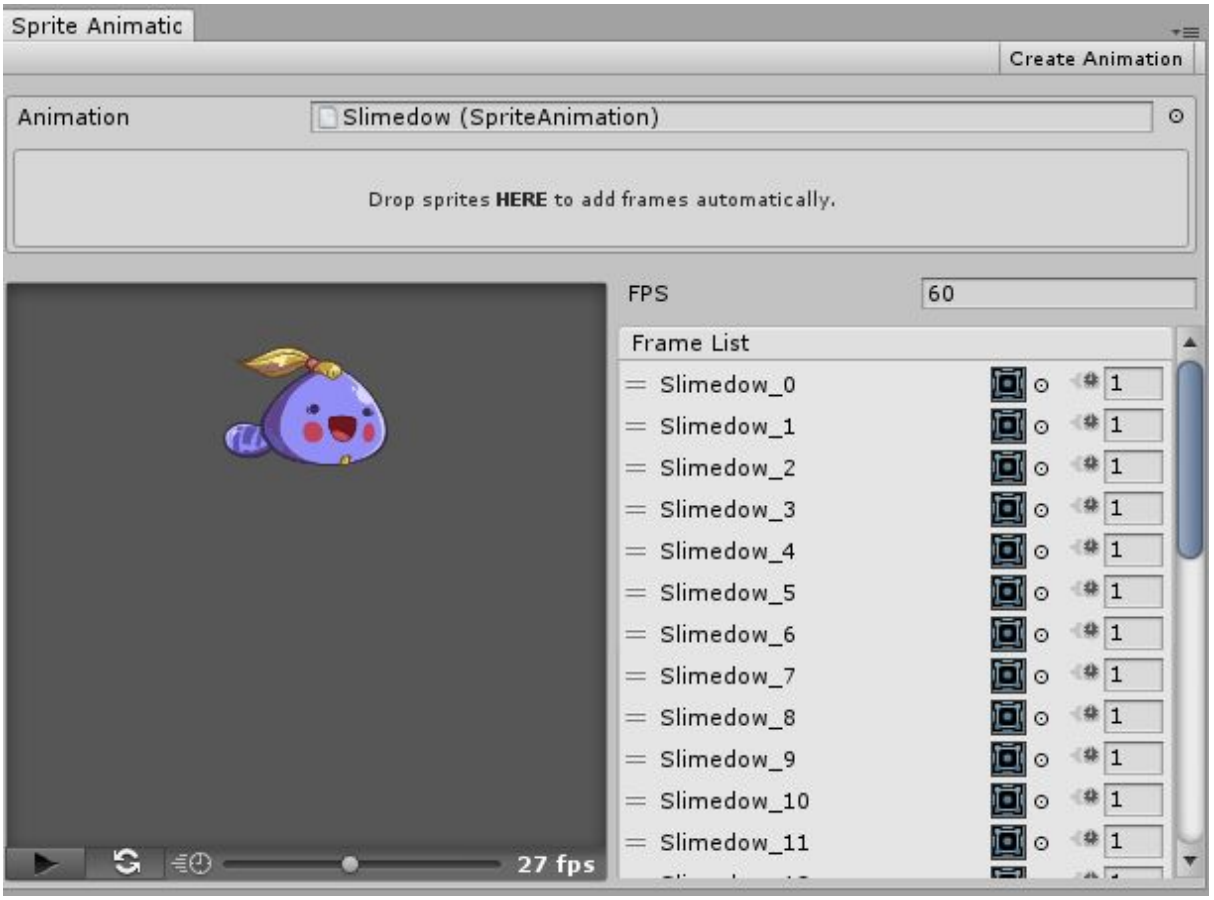
Thank you for your purchase! In this document you have all the info needed to use the plugin. If you have any questions don't hesitate to send me a mail to help@elendow.com and I'll try to help you out :)

You can also post your suggestions or questions in the Unity Forums thread.

Creating an animation

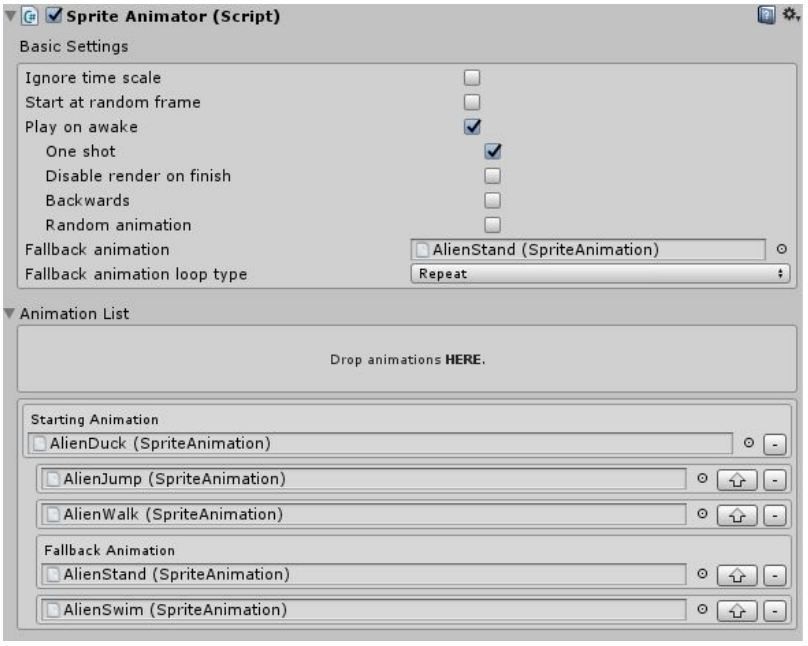
Use the animation editor to create new animation files. You can open it selecting Sprite Animation Editor on Elendow Tools tab.

- Give a name to the animation. This will be also the asset name. This name will be the one used to play the animations.
- Select the folder to save.
- Select the framerate of 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 or delete the frames.
- 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 TimeScale** will set the animation to ignore the game TimeScale.
- **Start At Random Frame** will set the animation to start at random frame when “Play” is called.
- **Play on Awake** will start playing when the object awakes.
    - **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
    - **Disable Renderer 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.
    - **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.
- **Fallback animation** will set an animation to play when theres no other animation playing.
    - **Fallback animation loop type** will set the loop type for the fallback animation.
- **Animations** is a list with all the animations.
    - You can drop your animation on the box to add them to the list
    - The arrow button will move the animation to the first place
    - The “-” button will remove the animation from the list.
    - The first animation will the one to play when **Play on Awake** is enabled.



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 and fill the animations list with the animations you want. 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 (string animationName, bool oneShot = false, bool backwards = false, LoopType loopType = LoopType.repeat)** plays the animation infinite times if oneShot = false with the selected loop type, only one time if true, forward if backwards = false and backwards if it's true.
    - If the animation is the same that is playing, nothing will happen but the oneShot attribute will update.
    - If the animation is the same that was playing but its not playing now, the animation will Reset and Resume and the oneShot attribute will update.
    - If the animation is different, it will play the new animation from the start.
- **Play (bool oneShot = false, bool backwards = false, LoopType loopType = LoopType.repeat)** plays the first animation of the animation list.
- **Play (SpriteAnimation animation, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)** plays the specified animation, this animation doesn't has to be on the animations list of the animator, yay!
- **PlayRandom (bool playOneShot = false, bool backwards = false, LoopType loopType = LoopType.repeat)** plays a random animation from the animation list.

- **PlayStartingAtFrame (string name, int frame, ool playOneShot = false, bool backwards = false, LoopType loopType = LoopType.repeat)** plays an animation starting at the specified frame.
- **PlayStartingAtFrame (int frame, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)** plays the first animation of the animation list starting at the specified frame.
- **PlayStartingAtFrame (SpriteAnimation, animation, int frame, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)** plays the first animation of the animation list starting at the specified frame.

- **PlayStartingAtTime (string name, float time, ool playOneShot = false, bool backwards = false, LoopType loopType = LoopType.repeat)** plays an animation starting at the specified time (in seconds).
- **PlayStartingAtTime (float time, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)** plays the first animation of the animation list starting at the specified time (in seconds)..
- **PlayStartingAtTime (SpriteAnimation, animation, float time, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)** plays the first animation of the animation list starting at the specified time (in seconds).

- **PlayStartingAtNormalizedTime (float normalizedTime, bool playOneShot = false, bool playBackwards = false, LoopType loopType = LoopType.repeat)** Plays the first animation of the animation list starting at the specified normalized time (between 0 and 1).
- **PlayStartingAtNormalizedTime (string 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).
- **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 current animation.
- **Reset ()** restarts the animation (playing or not) to its initial state. If the animation is not playing, the restart will be applied only when it start playing again.
- **Stop ()** stops the current animation.
- **StopAtFrame(int frame)** 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.
- **SetFallbackAnimation(string animation, LoopType loopType)** Sets the fallback animation to play and its loop type.
- **RemoveFallbackAnimation()** Removes the fallback animation
- **SetActiveRenderer (bool active)** enable/disables the renderer.
- **FlipSpriteX (bool flip)** flips the sprite on the X axis.
    - Not working on UIAnimator yet.
- **FlipSpriteY (bool flip)** flips the sprite on the Y axis.
    - Not working on UIAnimator yet.
- **AddCustomEvent (int frame)** adds an event to a specific frame of the first animation of the animation list and returns it.
- **AddCustomEvent (string animation, int frame)** adds an event to a specific frame of an animation and returns it.
- **AddCustomEvent (SpriteAnimation animation, int frame)** adds an event to a specific frame of an animation and returns it.

- **AddCustomEventAtEnd ()** adds an event to the last frame of the first animation of the animation list and returns it.
- **AddCustomEventAtEnd (string animation)**adds an event to the last frame of an animation and returns it.
- **AddCustomEventAtEnd (SpriteAnimation animation)**adds an event to the last frame of an animation and returns it.
    - The event subscriber must have this structure **MethodName(BaseAnimator caller){}**
- **GetCustomEvent (int frame)** returns the event of the first animation of the animation list at the specific frame. Returns null if there's no event on that frame and animation.
- **GetCustomEvent (string animation, int frame)** returns the event of the animation at the specific frame. Returns null if there's no event on that frame and animation.
- **GetCustomEvent (SpriteAnimation animation, int frame)** returns the event of the animation at the specific frame. Returns null if there's no event on that frame and animation.
- **GetCustomEventAtEnd ()** returns the event of the first animation of the animation list at the last frame.
- **GetCustomEventAtEnd (string animation)** returns the event of the animation at the last frame.
- **GetCustomEventAtEnd (SpriteAnimation animation)** returns the event of the animation at the last frame.
- **SetRandomDelayBetweenLoops(float min, float max)** sets a random delay between loops. The animation will stay at the last frame, but you can use **DisableRenderOnFinish** to avoid this.
- **SetDelayBetweenLoops(float delay)** sets a fixed delay between loops. The animation will stay at the last frame, but you can use **DisableRenderOnFinish** to avoid this.
- **SetAnimationTime(float time)** sets the animation time to the specified time in seconds, updating de sprite to the correspondent frame at that time.
- **Initialize(bool playOnAwake, List<SpriteAnimation> animations, string startAnimation)** manually initialize the animator. Useful and **NECESSARY** if the animator was instanced on runtime.
- **UseAnimatorFPS(int frameRate)** sets the animator FPS overriding the FPS of the animation.
- **UseAnimationFPS()** sets de animator FPS to the current animation FPS.

Animator Properties

- **IsPlaying { get; }** returns true if the animation is playing and false if not.
- **CurrentAnimation { get; }** returns a string with the current animation name.
- **CurrentFrame { get; }** returns the current frame of the animation.
- **DisableRenderOnFinish { set; }** sets the disableRenderer attribute. This will disable the renderer when the animation ends.
- **RandomAnimation { set; }** if true the animator will get a random animation after every loop cycle

- **StartAtRandomFrame { set; }** if true the animator will start the animations at a random frame instead of the first one. Cool if you want to desynchronize animations.
- **StartAnimation { get; set; }** the animation to play with Play On Awake active and Random Animation disabled.
- **CurrentFrameRate { get; }** the current FPS of the animator (it could be the animation FPS or an overrided FPS)
- **CurrentAnimationTime { get; }** the current time in seconds of the playing animation

Animator Events

- You can suscribe to the animation events using the **AddListener(Listener)** method of the UnityEvent class.
- **onFinish** calls when the animation reach the last frame.
- **onPlay** calls when the animation starts playing.
- **onStop** calls when the animation is forced to stop.
- You can add an event to a specific frame of an animation using the method **AddCustomEvent**(string animation, int frame).
    - The event subscriber must have this structure **MethodName(BaseAnimator caller){}**
    - Ex: animation.AddCustomEvent("Walk", 3).AddListener(StepFrame). Now on the frame 3 of the animation "Walk" the method StepFrame will be called.

Animation Methods

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

Animation Properties

- **Name { get; }** Returns the name of the animation file.
- **FPS { get; set; }** The FPS of the animation.
- **FramesCount { get; }** Frames on this animation.
- **Frames { get; set; }** The list of the frames in this animation.
- **FramesDuration { get; set; }** The list of the durations (in frames) of the frames in this animation.
- **AnimationDuration { get; }** The total duration of the animation (in frames)

Editor Utils

- **[SpriteAnimationField]** : Use this attribute to transform a string field in a animation list field. This is useful to avoid human errors when setting animations from the inspector.

  Keep in mind that this attribute needs a Sprite Animator in the object containing the script OR a child of it. If multiple Sprite Animators are found, only the first one is used.