Spryt Animator Pro Documentation

Spryt developed by Bluish-Green Productions, for help, support, and feedback, please contact Support

Table of Contents (Hyperlinked)

Features

Simplified yet powerful Sprite Animation and Manipulation
Spryt Animator Pro allows you to breathe new life into single-frame sprites and add some extra appeal to multi-frame animations through simple image manipulation. In addition, Spryt Animator Pro provides a quick and easy way to manipulate image properties via script. It can even completely replace Mecanim to remove the need for multiple Animators and Animations cluttering up your project.
Features:

- **Bring Static Images to Life**: Create *squishy Sprites* and *bouncy UI Images* through integrated *Sine Effects*.
- **UI Compatible**: Works on both *Sprite Renderer* and *UI Images*.
- **Simplified Syntax**: Manipulate image properties at runtime without cumbersome Method calls or GetComponent calls (*sprite.Alpha = 0.9f;*)
- **Replace or Complement Mecanim**: No need for creating Animators and Animations for each animated asset, though you can still take advantage of Spryt animation effects while using Mecanim
- **No Strings**: Do away with unreliable *String references* when switching between animations
- **Control Playback**: Play, Pause, Reverse, PlayOneShot *animation Methods*
- **Animation End Event**: Trigger methods using the *Animation End* Event
- **Sine Effects**: Includes "*Sine" utility class* for use beyond Sprite manipulation
- **Fully Documented***:* Scripts include "summary" tags for *Intellisense* in Visual Studio

Getting Started

Compatibility:

- **.NET Compatibility**: To ensure compatibility with Spryt Animator Pro scripts, you will need to use a newer standard of .NET in Unity. It is recommended to do this before Importing the Package to avoid any Compile errors, but it can be done after importing the package. The way this is done varies based on which Version of Unity you are using:
    - **For Unity 2017x**: "**Edit > Project Settings > Player > Other Settings > Configuration > Scripting Runtime Version**" should be set to "**Experimental (.NET 4.6 Equivalent)**". Making the change will require a restart of Unity.
    - **Unity 2018 onwards**: "**Edit > Project Settings > Player > Other Settings > Configuration > Ape Compatibility Level**" should be set to "**.NET 4.x**"
- **Complete Package**: *Unity Version 2019.2.1f1*
    - Due to errors resulting in exporting Asset packages from Unity Version 2019.2.1f1, the *SampleScene* can only be opened in this version or later.
- **Core Scripts**: *Unity Version 2017.1.5f1*
    - The core 7 Scripts which make up Spryt Animator Pro are tested to be compatible with Unity Version **2017.1.5f1** (at the date of writing, the earliest version of Unity you could install from Unity Hub).
    - The reason earlier versions of Unity are not supported is because Unity 2017.1 is the earliest version which supports .NET 4x, which is required for the reasons listed above.

Setup

A video guide is available for setting up Spryt Animator Pro.

Importing Spryt Animator Pro

1. **Ensure you have a compatible version of Unity**: Please use Unity Hub to install a version of Unity equal to or later than 2017.1.5f1. Please see above notes on Compatibility to see which limitations may exist depending on which version of Unity you are using.
2. **.NET Compatibility**: To ensure compatibility with Spryt Animator Pro scripts, you will need to use a newer standard of .NET in Unity. It is recommended to do this before Importing the Package to avoid any Compile errors, but it can be done after importing the package. The way this is done varies based on which Version of Unity you are using:
    A. **For Unity 2017x**: "**Edit > Project Settings > Player > Other Settings > Configuration > Scripting Runtime Version**" should be set to "**Experimental (.NET 4.6 Equivalent)**". Making the change will require a restart of Unity.
    B. **Unity 2018 onwards**: "**Edit > Project Settings > Player > Other Settings > Configuration > Ape Compatibility Level**" should be set to "**.NET 4.x**"
3. **Import the Package into Unity**: With the project open in the Unity asset store, click the Import button.
4. **Import everything, or just the Scripts**: The "Import Unity Package" window should now be open and you can decide if you want to import everything, or exclude the contents of the "Example" folder. For first time users, it is recommended to import everything using *Unity Version 2019.2.1f1*, but experienced users can exclude the Example folder and import into any version of Unity as old as *Version 2017.1.5f1*. Once you have made your choice, select "Import"
5. **Errors? Don't Panic**: You may get as many as 10 Compile errors if you did not update Unity to a new version of .NET, as noted above. If you are getting errors unrelated to .NET compatibility, please copy them and submit them to a new request for Support.

Structure:

Spryt Animator is composed of 7 Scripts, 3 of which are Scriptable Objects. Implementation is as easy as attaching either SprytSingle or MultiSpryt to a GameObject. Which of the two you use will depend on what you are trying to accomplish.

- **Spryt**: A *Scriptable Object* which is a List of Sprites. Used by MultiSpryt to switch between sets of frames. Once created, a "Spryt" can be used throughout the project, or even between projects.
- **BaseSpryt**: The parent class of SprytSingle and MultiSpryt which has all Methods and Properties.
    - **SprytSingle**: A streamlined derived class which allows you to drop Sprite frames directly onto a GameObject without need for the Spryt Scriptable Object.
    - **MultiSpryt**: A derived class specially designed for manipulating multiple sets of animated frames through the Spryt Scriptable Object.
- ***Spryt Controller***: You will need to create a Script to handle manipulating properties, switching between Spryts at runtime, and activating Sine Effects. This "controller" functionality can be appended to an existing class, (like a Character controller). The details of this controller will vary depending on whether you are using SprytSingle or MultiSpryt and are covered in their respective Setup documentation below. In addition, there are multiple "controller" scripts used in the Sample Scene provided with Spryt Animator Pro which can be used as a reference for implementation details.
- **Sine**: Sine is a Utility class which can be used independently of all other Spryt systems. BaseSpryt makes heavy use of it, and therefore it should not be deleted even if you do not intend to use Sine Effects in your Project.
    - **SineData**: A *Scriptable Object* which holds all the data for an instance of the Sine Class. This can be used to create a Sine effect which can be re-used throughout the project, or even between projects.
    - **SSineData (Short for *SprytSineData*)**: A *Scriptable Object* which holds all the data for the x / y, x / y Scale, and Angle Sine effects used by Spryt Animator Pro. This can be used to create a batch of Sine effects which can be re-used throughout the project, or even between projects.

Importing Sprites into Unity

Both Spryt and MultiSpryt make use of Unity Sprites, so this documentation will provide a basic guide on importing your first sprites for use with the Spryt Animator Pro Scripts. For more in-depth information on the Unity Sprite Editor, please see the Official Documentation.

1. **Drop in the Asset**: Either drag-and-drop the image assets you wish to use for your sprites into your Project Asset folder, or right-click within the Asset folder and select "Import New Asset...". You may wish to put the assets into a sub-folder for sake of organization.
2. **Slice your Sprites**: If you are importing a Sprite-sheet (a single image asset containing multiple Sprites) you will need to slice it into individual frames which Spryt Animator Pro can use. If you are importing each frame as a separate image asset, you do not need to slice your sprites, and can move on to "Common Setup" below.
    a) **Set Sprite Mode**: With the image asset selected, and its properties open in the Inspector, switch the "*Sprite Mode*" to "*Multiple*"
    b) **Apply**: Click "*Apply*" down at the bottom of the properties.
    c) **Open the Sprite Editor**: Click the "Sprite Editor" button to open the selected image asset in a new window.
    d) **Slice**: Click "Slice" along the top bar of the Sprite Editor Window
    e) **Select Slice Type**: You can try your luck with the Automatic Slicing option, or pick from either "Grid by Cell Size" or "Grid by Cell Count" for more control. Either option will work for a well-made Sprite-sheet, but note that most Sprite-sheets I have encountered (especially fan-made sprite-rips) are not well made and will require lots of pain-staking effort to extract the sprites. It is beyond the scope of this documentation to assist in extracting such sprites, but you can refer to the FAQ for a guide on creating well-made Sprite-sheets.
    f) **Enter Settings**: Once you have selected the Type of Slicing you will use, enter either the particular details related to the sprite sheet you are slicing, then click the "Slice" button.
    g) **Apply**: The Sprite sheet should now be sliced, allowing you to click individual frames. Once satisfied with the results, click "Apply" near the top-right of the Sprite Editor window.
    h) **Close**: You may now close the Sprite Editor window and proceed to "Common Setup" below.

Common Setup

- **For Sprite Renderer**: To add an animated instance to your project, select "**GameObject** > **2D Object** > **Sprite**" (or simply right-click in the Hierarchy).
- **For a Canvas Image**: To add an animated instance to a UI Canvas in your project, select "**GameObject > UI > Image**" (or simply right-click in the Hierarchy).
    - *Note that all UI images must be the child of a Canvas instance*.
- **Assign a default Sprite**: Whether you are using a Sprite Renderer or a UI Image, you can now assign a default Sprite which will be displayed in the Scene view.
    - **Visual Preview**: Note that this can be a preview image specifically intended for use while the game is not running, as both SprytSingle and MultiSpryt have the capacity to replace this default Sprite.
    - Also note that this step is not *strictly* necessary as Spryt Animator will immediately replace the selected Sprite with the first frame supplied to it as the game runs. However, without this default sprite, you may have trouble seeing the instance in the Scene view tab unless you have some other way of visually identifying it.
- **Optional (child-image)**: Since Sine Effects actually change the size and shape of the GameObject on which they are used, you may want to make the Spryt itself a child of the GameObject you are using. This method is used on the Player in the Sample Scene included in Spryt Animator Pro.
- **Choose based on need**: This completes Common Setup, you may now choose either SprytSingle or MultiSpryt to attach to the GameObject based on whether you need to use a single set of frames for one, looping animation, or if you will switch between multiple sets of frames for something more complex like a player character.
    - **Note**: The chosen Script must be attached to the same GameObject which has the image you intend to animate. You cannot place the Scripts on a GameObject above or below the image in the hierarchy.
- **Re-use frames**: In certain circumstances, an animation may use the same frame multiple times throughout its duration. Since Spryt Animator Pro deals with simple Lists of Sprites, this can easily be done by dropping the repeated frame(s) into the List wherever they are needed.

SprytSingle Setup

SprytSingle requires incredibly little actual setup, especially if you only intend to use it for playing a single, looping animation.

- **Add Component**: Begin by adding the "SprytSingle" Component to the GameObject. A quick way to do this is to type "yts" into the text-field, (or "Single" if that is easier to remember) unless you have other Scripts in your game which include this set of letters in their names.
- **Add Frames**: Now, you can simply drag-and-drop the frames for the animation directly onto the "Sprite Frames" property of SprytSingle, although you may wish to *Lock your Inspector* in order to make this easier to do so.
    - **Multi-Drag**: You can drag all the frames of an animation onto the Component at once by holding shift, selecting all frames, then clicking and dragging. If the frames are not consecutive, you can use the Control key to modify which frames are selected.
    - **See frames listed**: If done correctly, you should be able to expand the "Sprite Frames" List and see the frames listed as elements, with the Size of the List now displaying the number of Elements.
    - **Unlock**: If you chose to Lock your Inspector, now would be a good time to Unlock it before moving on.
- **That's it (basically)**: At this point, SprytSingle is ready to go, although you may wish to tweak the Speed property of the animation based on how many frames you are using.
    - You can also select a SprytSineData object to apply Sine Effects which will animate the GameObject, but the particulars of this are covered as part of the "Sine Scripting" documentation under the heading "Sine Effect Setup"
- **Advanced Control**: You will need to create a Script to handle manipulating properties and activating Sine Effects at runtime. This "controller" functionality can be appended to an existing class (like a UI Button controller).
    - **SprytSingle Reference**: In order to manipulate the Properties of SprytSingle, you will need a reference to it. For instance:
      ```
      public SprytSingle sSpryt;
      ```
        - This line declares a reference to an instance of "SprytSingle" named "sSpryt" which will likely be populated using the Inspector. You could also make the field private and use GetComponent in Start or Awake.
    - **Manipulating the SprytSingle Component**: Once you have a reference to the SprytSingle Component, you can manipulate its properties, hook into its Animation End Event, etc. Here is a simple example where setting the alpha value of the sprite:
      ```
      sSpryt.Alpha = 0.9f;
      ```
    - **In greater Depth**: For a more detailed example on how to manipulate the properties of a SprytSingle Component, see the various sample Button controller Scripts provided in the sample Scene.
        - For a complete look at the Properties and Methods available, see the section "*Spryt Animator Pro Scripting*" below.

MultiSpryt Setup

- **Create Spryt(s)**: MultiSpryt makes use of an intermediary asset; the "*Spryt*" Scriptable Object to switch between sets of frames. To create a *Spryt*, select "**Assets > Create > Spryt Animator Pro > Spryt (Frame List)**" (You can also right-click in any Asset Folder in the Project tab to create a Spryt).
- **Populate the Spryt**: Once the Spryt asset has been created, you can add frames to it (just as you would add frames directly to a SprytSingle). You may wish to *Lock your Inspector* in order to make this easier to do so.
    - **Multi-Drag**: You can drag all the frames of an animation onto the Spryt object at once by holding shift, selecting all frames, then clicking and dragging. If the frames are not consecutive, you can use the Control key to modify which frames are selected.
    - **See frames listed**: If done correctly, you should be able to expand the "Frames" List and see the frames listed as elements, with the Size of the List now displaying the number of Elements.
    - **Unlock**: If you chose to Lock your Inspector, now would be a good time to Unlock it before moving on.
- **Add Component**: Next, add the "MultiSpryt" Component to the GameObject you intend to animate. A quick way to do this is to type "ryt" into the text-field, since between "SprytSingle" and "MultiSpryt", MultiSpryt shows up first alphabetically.
- **Add a Spryt**: Once the MultiSpryt Component has been added, you should assign a Spryt to its Index property. This will be the very first set of animated frames Spryt Animator will display.
    - **Set Speed**: At this point, you may wish to tweak the Speed property of the animation based on how many frames you are using.
    - **Sine Effects**: You can also select a SprytSineData object to apply Sine Effects which will animate the GameObject, but the particulars of this are covered as part of the "Sine Scripting" documentation under the heading "Sine Effect Setup"
- **Swapping Spryts**: You will need to create a Script to handle manipulating properties, switching between Spryts at runtime, and activating Sine Effects. This "controller" functionality can be appended to an existing class, (like a Character controller) which could sit on the *same* GameObject, or elsewhere, depending on the nature of the manipulation.
    - **MultiSpryt Reference**: In order to manipulate the Properties of MultiSpryt, you will need a reference to it. For instance:
      ```
      public MultiSpryt mSpryt;
      ```
        - This line declares a reference to an instance of "MultiSpryt" named "mSpryt" which will likely be populated using the Inspector. You could also make the field private and use GetComponent in Start or Awake.
    - **Spryt References**: In order to switch between Spryt assets, you will need a few references to Spryt objects in your controller script:
      ```
      public Spryt sStand;
      public Spryt sWalk;
      ```
        - The above illustrates an example found in the "*Player.cs*" script in which a total of 5 separate Spryt object references are declared.
    - **Swapping Sprites**: Switching from one Spryt object to another can be done by assigning to the Index Property of MultiSpryt:
      ```
      mSpryt.Index = isRunning ? sWalk : sStand;
      ```

        - The above illustrates an example found in the "*Player.cs*" script which uses a Ternary Expression to switch between Standing and Walking based on the "*isRunning*" boolean variable.
    - **In greater Depth**: For a more detailed example on how to manipulate the properties of a MultiSpryt Component, see the "Player.cs" Script provided in the sample Scene.
        - For a complete look at the Properties and Methods available, see the section "*Spryt Animator Pro Scripting*" below.

Example Overview:

One of the best ways to familiarize yourself with the use of Spryt Animator Pro is to try out things in the Sample Scene and look through the sample Scripts.

Naming Convention:

The naming convention used in the Example scripts is based on the naming conventions I used in my years of developing with GameMaker. You may find it useful to adhere to the following conventions, or perhaps establish your own conventions.

- **SprytSingle / MultiSpryt instances**: You will see instances in this documentation and in the Scripts themselves where I have used “mSprite” for MultiSpryt, but in my personal scripts, I simply use the name “sprite” for both instances of MultiSpryt or SprytSingle for the following reasons:
    - Both classes have nearly identical Properties, Fields, and Methods
    - You would only (reasonably) be using one or the other at a time
    - “Spryt Animator” is an interface to manipulate the GameObject’s “Sprite”
        - ***Note****: Not a literal C# Interface!*
- **Spryts**: Each Spryt reference is named with a lower-case letter “s” followed by the name of the animation:
    - `public Spryt sWalk;`
- **SineData / SprytSineData**: I regard these resources to be “animation” data, so I used a lower-case letter “a” followed by the name of the “animation”. This way if there is an overlap with a Spryt (there is both a Spryt named “Jump” and SSineData named “Jump” there is no conflict in the names).
    - `public SSineData aJump;`

Features in the Sample Scene

Here are the explanations available in the Demo Scene:

- **Player**: Sample MultiSpryt character. Move with A/D, Space to Jump, Crouch with S. Touch the Skull to trigger death, click "Respawn Player" if you die or fall off-screen. Uses multiple Spryts, but only the walking animation has multiple key-frames, other animation is handled through Sine Effects. Switching Spryts and Sine FX is controlled through the "Player" Script, which also makes use of the Animation End Event to play a step sound at the end of each cycle of the animation.
- **Death**: Demonstrates use of Sine Effect Events. The player dies if it touches the Skull, switching to a death prefab which plays a shudder Sine Effect. An Event is fired at the end of the Shudder triggering a fade-out with another Sine Effect. Finally, another Event is fired when the fade-out concludes, removing the death prefab.
- **Skull**: Demonstrates compatibility with Mecanim. The skull has an Animator Component to handle switching its Keyframes, but uses SprytSingle to take advantage of Sine Effects. Simply assign SprytSingle to a GameObject and leave its “Sprite Frames” List empty!
- **Hello**: The Hello Box demonstrates the PlayOneShot capability of Spryt. This could also be achieved by hooking into the Animation End Event and Pausing or setting Speed to 0, but this provides an expedited way of achieving the same effect.
  The Hello box is actually a child of the Canvas, and so technically doesn’t belong in the Sprite Renderer demo zone, but space is limited when you try to fit everything in one Scene.
- **Buttons**: The UI Buttons in the Scene demonstrate Spryt’s integration with UI Images. For instance, the Respawn button uses Pointer Events to display unique Sine Effects on pointer-enter, click, and exit. They use SprytSingle with speed set to zero, manually switching between frames depending on pointer position. You can also look at the “ExplanationButton” Script to for how to handle Pointer Events.
- **Arrow**: The Arrow has a simple script attached to it which has it aim at the mouse by manipulating the Spryt’s angle property. It has a continuous sine effect playing to manipulate the x/y scale of the image. You can see that with only a single Frame, you don’t even need to specify any frame data on SprytSingle, it will pull the sprite from the Sprite Renderer.
- **Canvas**: The canvas to the right has numerous buttons which directly manipulate the Properties of the Spryt on display. While the sliders in this Canvas have bounds, this is not the case when manipulating these Properties directly through Script. The “Sample Controller” Script on the Image Container has a few methods on it which serve as an in-between for the Spryt and the Canvas buttons.

FAQs (Frequently Asked Questions)

- **I'm getting Compile Errors after importing the asset into my project**
  - **.NET Compatibility**: To ensure compatibility with Spryt Animator Pro scripts, you will need to use a newer standard of .NET in Unity. It is recommended to do this before Importing the Package to avoid any Compile errors, but it can be done after importing the package. The way this is done varies based on which Version of Unity you are using:
  - **For Unity 2017x**: "**Edit > Project Settings > Player > Other Settings > Configuration > Scripting Runtime Version**" should be set to "**Experimental (.NET 4.6 Equivalent)**". Making the change will require a restart of Unity.
  - **Unity 2018 onwards**: "**Edit > Project Settings > Player > Other Settings > Configuration > Ape Compatibility Level**" should be set to "**.NET 4.x**"
- **The Example Scene doesn't import properly**
  - Due to an issue exporting Asset Packages from *Unity Version 2019.2.1f1*, the *Example Scene* can only be properly viewed in this version of Unity or later. The Scripts that form the core of Spryt Animator Pro will function as far back as *Unity Version 2017.1.5f1*
- **My Sprites are tiny!**
  - In the *Sprite Import Settings*, change the "*Pixels Per Unit*" to a number smaller than "100" (the exact size will depend on your artwork).
- **My Sprites are blurry!**
  - In the *Sprite Import Settings*, change the "*Filter Mode*" to "*Point (no filter)*"
  - You may also need to change "*Compression*" to "*None*"
- **How do I use Spryt Animator with Unity's Mecanim?**
  - Simply assign *SprytSingle* to a GameObject and leave its "Sprite Frames" List empty, this will allow you to use Sine Effects and most other Properties of the Spryt Animator system, but you will not have control over playback or any of the Animation methods (since Mecanim is now in charge of this).
- **How do I use Spryt Animator Pro on a UI Image?**
  - Spryt Animator Pro works identically on UI Images as it does with regular Sprite Renderer images. Simply attach either SprytSingle or MultiSpryt to the GameObject.
  - As an added bonus, if Spryt Animator Pro detects that it is being used in a Canvas, you can use the "*rect*" property for easy access to the RectTransform Component of the image.
- **I'm not going to use Sine Effects, can I delete the Sine script from my project?**
  - The "*Sine*" class is used by BaseSpryt, and so cannot be removed.
- **What's the best way to make a Sprite-Sheet for use in Unity?**
  - This ultimately depends on what you're using it for. Here are some quick rules-of-thumb. Note that following these steps will make it *easy* to import / use sprites, but it will *not be very efficient* (your Sprites will end up with lots of blank space).
  - **Maximise Dimensions**: It is best for each image in a sheet to occupy the maximal dimensions required by the animation.
    - Even if one frame could be smaller than the others, pad it with transparent pixels so that they all match width / height.
    - For example, if one frame is 12x16 while another is 16x16, add 4 pixels to the width of the first frame.

- **Anchor Positions**: If the sprite-sheet is of a character action, the character should maintain its position relative to other frames.
    - If a character is taking a step forward as part of a sword swing, you will likely want to use the foot that doesn't move, acting as a sort of "anchor" between frames.
- **Buffer your Tile-sheets**: If you're working with tile-sheets (for use in creating a tiled background / foreground) buffer your tile-sheets. To understand why you should do this, you can read this documentation (yes, it's for an old version of GameMaker, but it totally applies!)

Spryt Animator Pro Scripting

Fields:

- **Speed**: The speed at which the frames are cycled.
- **isPaused**: Whether the animation is paused or playing. You can use Pause / Resume / Toggle Methods to Set this field.
- **SprytSineData**: A set of Sine Data used to initialize all 5 Sine Effects on the Spryt
- **Hidden in Inspector**:
    - **rect**: A reference to the Spryt's RectTransform for use in manipulating its position if it exists in the context of a Canvas
    - **sineX**: A Sine Effect which changes the Spryt's x Position.
    - **sineY**: A Sine Effect which changes the Spryt's y Position.
    - **sineScaleX**: A Sine Effect which changes the Spryt's x Scale.
    - **sineScaleY**: A Sine Effect which changes the Spryt's y Scale.
    - **sineAngle**: A Sine Effect which changes the Spryt's Rotation.

Properties:

- **Frame**: The current index of the frames. The actual frame being displayed is this value floor-rounded.
- **Count**: The number of frames in the current Spryt Index. Set automatically when a new Spryt is assigned to the instance.
- **Visible**: Controls whether the Renderer is enabled or not. When this property is false, the Update method is not executed (the frame index and Sine Effects are not updated).
- **Alpha**: Controls the opacity of the Sprites between [0f: Transparent] to [1f: Opaque]. In contrast to Visible, setting the Alpha to 0f will still update the frame index.
- **Color**: Sets the Color on the Renderer. This does not modify the Alpha of the Spryt; the Alpha Property must be set independently.
- **x**: Independent control of the x Position of the Spryt.
- **y**: Independent control of the y Position of the Spryt.
- **Angle**: Changes the rotation of the Spryt between -180f and 180f.
- **xScale**: Independent control of the x Scale of the Spryt.
- **yScale**: Independent control of the y Scale of the Spryt.

The AnimationEnd Event:

AnimationEnd(): An Event which fires when the animation reaches the end of its frame count, forward or backwards.

- **Implementation**: Here is an example of implementing the AnimationEnd Event as seen in the “Player.cs” Script provided with the Example scripts:

```
public MultiSpryt mSprite;


private void OnEnable() {
//Subscribe to the Animation End Event generated by mSprite
        mSprite.OnAnimationEnd += Sprite_AnimationEnd;
    }


    private void OnDisable() {
//Unsubscribe from the Animation End Event generated by mSprite;
important to avoid memory leaks!!
        mSprite.OnAnimationEnd -= Sprite_AnimationEnd;
    }


    private void Sprite_AnimationEnd(object sender, Spryt spryt) {
//Each Animation End event passes along the Spryt which was playing
which you can compare against so unique things can occur depending on
which Spryt ended
        if (spryt == sWalk) {
    //When the walk animation plays through, play a footstep sound
            audioSource.Play();
        }
    }
```

Animation Methods:

- **Pause()**: Pauses the animation, freezing the Spryt on the current frame.
    - **Pause(*int* frame)**: Pauses the animation on a specified frame.
- **Resume()**: Resumes playing the animation and disables "playOneShot" behaviour.
- **Toggle()**: Pauses the animation or Resumes it depending on its current state
- **Last()**: Pause on the Last frame
- **First()**: Pause on the First frame
- **Reverse()**: Inverts the animation speed. If the animation is on the first or last frame, it jumps to the end or beginning.
- **PlayOneShot(*bool* pauseOnLastFrame)**: Plays the animation once through, then pauses it. Automatically jumps to the first frame if speed is positive, or the last frame if speed is negative. Also automatically unpauses the animation.
    - **pauseOnLastFrame**: When the animation ends, should it stop on the last frame or cycle back to the beginning?

Sine Effect Helper Methods:

- **ActivateSine(*SineType* sineType, *bool* max)**: Allows for independent activation of any of the 5 Sine effects on the Spryt
    - **sineType**: Which of the 5 Sine Effects to activate.
    - **max**: Set to true for Sine Effects which are meant to play in reverse.
- **ActivateSinePosition()**: A shortcut method to Activate only the x / y Position Sine Effects on the Spryt.
- **ActivateSineScale()**: A shortcut method to Activate only the x / y Scale Sine Effects on the Spryt.
- **ActivateSineRotation()**: A shortcut method to Activate only the rotation Sine Effect on the Spryt.
- **ActivateSineAll()**: A shortcut method to Activate all 5 Sine Effects on the Spryt.
- **SineSetData(*SSineData* ssData, *bool* activateAll)**: Sets all 5 Sine Effect on the Spryt to a new set from the supplied Spryt Sine Data.
    - **ssData**: A Spryt Sine Data Scriptable Object which has all the data to replace all 5 Sine Effects on the Spryt.
    - **activateAll**: Whether or not to immediately activate the newly supplied effects.
- **SineSetData(*SineData* sineData, *SineType* sineType, *bool* activate)**: Allows replacement of individual Sine Effects on the Spryt.
    - **sineData**: A Sine Data Scriptable Object which has the data to replace 1 of the 5 Sine Effects on the Spryt.
    - **sineType**: Which of the 5 Sine Effects to update.
    - **activate**: Whether or not to immediately activate the newly supplied effect.

Utility Methods:

- **AssignFrames(*List<Sprite>* sprites)**: A publicly accessible Method which can be used to assign a List of Sprites to the internal List used by Spryt. This method is called each time a new Spryt is assigned to the Index of this instance.
    - **sprites**: A List of type Sprite, in the order the frames should be shown.
- **ResetSprite()**: Resets all properties of the Spryt to default values as defined on the GameObject in the Inspector.

- **Scale()**: Allows compound assignment of both the x / y Scale Properties simultaneously. The Scale method has the following overloads:
    - **Scale(*float* newScaleFloat)**: This single float will be applied to both the x / y Scale of the Spryt
    - **Scale(*float* xS, *float* yS)**: xS sets the xScale of the Spryt, while yS sets the yScale
    - **Scale(*Vector2* newScaleVector)**: The x component of this Vector will be applied to the Spryt's xScale, the y component of this Vector will be applied to the Spryt's yScale
- **RelativeScale(*float* rX, *float* rY)**: Allows for scale changes to be performed on the Spryt without directly changing its X / Y Scale properties. Especially useful for Sine Effect animations.
- **RelativePosition(*float* rX, *float* rY)**: Allows for position changes to be performed on the Spryt without directly changing its X / Y properties. Especially useful for Sine Effect animations.
- **RelativeRotate(*float* rZ)**: Allows for rotations to be performed on the Spryt without directly changing its Angle property. Especially useful for Sine Effect animations.
- **PositionOriginRecalibrate()**: Relative Position transformations are performed relative to a Spryt's origin, there are three variants of this method which allow the Origin to be modified:
    - **PositionOriginRecalibrate()**: This method treats the Spryt's current position as the new origin for the Spryt.
    - **PositionOriginRecalibrate(*Vector2* newOrigin)**: This method lets you specify a new Vector2 origin for the Spryt.
    - **PositionOriginRecalibrate(*float* originX, *float* originY)**: This method lets you specify a new origin for the Spryt using x / y floats.
- **GetName()**: Static Method which returns a string of the asset name of the supplied Scriptable Object

SprytSingle:

Structure:

- **Streamlined Single Spryt**: To be used on animated instance which will only ever use a single set of frames, SprytSingle provides a simple alternative to MultiSpryt which does not require the creation of Spryt objects.
- Instead, a List of Sprites can be added directly to the Component in the Inspector.

spriteFrames Field:

- **spriteFrames**: A List of Sprites used in the Start method of SprytSingle to define its animation.

MultiSpryt:

Structure:

- **Manage Multiple Spryts**: To be used on instances which will need to dynamically switch between sets of frames at runtime through script.
- Spryts can be assigned to the Index Property of MultiSpryt directly to switch between sets of frames.

Index Property:

- **Index**: Index accepts a Spryt Scriptable Object which is simply a List of Sprites. You can safely assign the same Spryt to Index multiple times and it will ignore subsequent attempts.

Methods:

- **Combo(*Spryt* newSpryt, *SSineData* sprytSineData)**: In a single method, assign a new Spryt, assign a new SprytSineData and Activate it
- **ToString()**: An override of ToString which returns a string of the current Spryt Index asset name.

Sine Scripting

One of the special features packaged into Spryt Animator Pro is the integration of Sine Effects which can be used to give life to static images through simple image manipulation. This is done through the use of the Sine utility class which utilizes Sine curve “easing” to provide satisfyingly smooth transitions. Each instance of the Sine class represents a variable whose position is tracked along a sine curve. You can assign the value of this variable to the x / y Position of a GameObject to make the instance shake back and forth, assign it to the scale of an instance to have it squash and stretch, or apply it to the rotation of an instance of have it oscillate. Spryt Animator Pro has an instance of the Sine class for each of these 5 properties to make it incredibly quick and easy to use, but this is only the beginning of what you can do with Sine Effects. For a more in depth look at the various ways you can use Sine-based effects in your Project, you can watch this promotional video in which several uses are highlighted.

Structure:

- **Sine**: Utility class used to create smooth transitions utilizing Sine curves.
    - **For independent use**: Sine can operate completely independently of Spryt, but the reverse is not true. In the name of making script-based sprite animation as easy as possible, BaseSpryt is heavily coupled to the Sine class.
- **SineData**: A Scriptable Object which holds all the data for an instance of the Sine Class.
    - **Re-Usable**: This can be used to create a Sine effect which can be re-used throughout the project, or even between projects.
    - **Overwrite**: It can be used independently of Spryt Animator systems, or it could be used to overwrite one of the five instances found in a Spryt instance.
- **SSineData**: A Scriptable Object which holds all the data for the x / y, x / y Scale, and Angle Sine Effects used by Spryt.
    - **Re-Usable**: This can be used to create a batch of Sine effects which can be re-used throughout the project, or even between projects.

Sine Effect Setup

- **Creating SineData**: To create a *SineData* object, select "**Assets > Create > Spryt Animator Pro > Sine Effect Data**" (You can also right-click in any Asset Folder in the Project tab to create a SineData Object).
    - **Populating SineData**: Filling out the data on SineData is identical to providing the data for an instance of the Sine Class. For more information on the fields of the Sine Class, refer to the "Fields" section below
    - **Utilizing SineData**: SineData holds the data of a Sine Class, it cannot itself be treated as an instance of the Sine Class (which you could simply declare and fill out in the Inspector). Use the new keyword and invoke a Constructor of the Sine Class to utilize the data in the SineData object, as is done in the "*PlayerDeadHandler.cs*" sample Script:

```
public SineData aFadeOut;
private Sine fadeOut;
private void OnEnable() {
        fadeOut = new Sine(aFadeOut);
}
```

- **Creating SSineData**: To create an *SSineData* object, select "**Assets > Create > Spryt Animator Pro > Spryt Sine Effect Data**" (You can also right-click in any Asset Folder in the Project tab to create an SSineData Object).
    - **Populating SSineData**: Filling out the data on SSineData is like providing the data for 5 instance of the Sine Class. For more information on the fields of the Sine Class, refer to the "Fields" section below.
    - **Utilizing SSineData**: To assign an SSineData object to either an instance of SprytSingle or MultiSpryt, you can drag the asset directly onto the "Spryt Sine Data" field of the Component.
        - **Play right away**: Any effects which have the "*effectActive*" boolean set to true will immediately begin to act upon the instance
        - **Await Activation**: Any effects which have the "*effectActive*" boolean set to false will not play until it is *activated*. This can be done through any of the "Sine Effect Helper Methods" documented in the "Spryt Animator Pro Scripting" section, or by referencing the sine variable directly:

```
mSpryt.sineX.Activate();
```

            - This will call the Activate method on the sineX effect variable belonging to an instance of a Spryt Class called "mSpryt"

Fields:

- **index**: The actual value which tracks the effect's progress. If it has a concept of duration, it will either approach zero or its maximum value. If not, it will be bounded between a range of 0 to 2*Pi.
- **divisor**: The divisor which controls how large of small a "slice of Pi" which is added to or subtracted from "index". A larger divisor results in a slower effect, a smaller divisor results in a faster effect.
- **magnitude**: A multiplier on the value of *index* to alter its range from the default -1 to +1
- **duration**: Measured in increments of Pi. A complete Sine curve has a duration of 2f while a quarter is 0.5f. If the effect is supposed to have limited duration (instead of continuing forever) set this to a non-zero number.
- **updateType**: If using the expedited SineUpdate method, this controls the Getter operation passed on the ref float.
- **effectActive**: A boolean to get or set whether the effect is currently playing. Set this to true if you want an effect to play immediately.
- **countDown**: If using the Sine's Update method, this controls whether the index will count up to its maximum value or down to zero.

The OnEffectEnd Event:

- **OnEffectEnd()**: An event fired when effects with limited duration elapse.
    - **Implementation**: Here is an example of implementing the OnEffectEnd Event as seen in the "*PlayerDeadHandler.cs*" Script provided with the Example scripts. There are two Sine *OnEffectEnd* Events being utilized in the Script, so for the sake of clarity, the usage of the SprytSingle sineX instance has been color-coded **orange**, while the stand-alone fadeOut Sine instance has been color-coded blue.

```
public SprytSingle sSprite;
public SineData aFadeOut;
private Sine fadeOut;
private void OnEnable() {
//Each of the 5 Sine effects on every Spryt has an associated event
you can tap into if it has a finite duration
        sSprite.sineX.OnEffectEnd += XSine_OnEffectEnd; //Subscribe to
the Effect End Event generated by the xSine of SprytSingle
        fadeOut = new Sine(aFadeOut); //New up "fadeOut" instance
based on the data provided in aFadeOut
        fadeOut.OnEffectEnd += FadeOut_OnEffectEnd; //Subscribe to its
EffectEnd
    }


    private void OnDisable() {
//Unsubscribe from the Effect End Events; important to avoid memory
leaks!!
        sSprite.sineX.OnEffectEnd -= XSine_OnEffectEnd;
        fadeOut.OnEffectEnd -= FadeOut_OnEffectEnd;
    }
private void XSine_OnEffectEnd(object sender, System.EventArgs e) {
//Activate the fadeOut effect. Since our Alpha starts at 1 and goes to
zero, we "maximize" the value of fadeOut instead of setting it to zero
        fadeOut.Activate(true);
    }


    private void FadeOut_OnEffectEnd(object sender, System.EventArgs
e) {
//When the fade-out effect completes, destroy the gameObject
        Destroy(gameObject);
    }
```

    - In this script, the *OnEffectEnd* Event of the SprytSingle sineX instance is used to trigger the fadeOut Sine effect, then the *OnEffectEnd* Event of fadeOut is used to Destroy the gameObject.

Constructors:

There are 3 Constructors for instances of the Sine Class:

- **Sine(*SineData* sineData)**: An expedited constructor which accepts a SineData Scriptable Object.
- **Sine(*bool* effectActive, *float* divisor, *float* magnitude, *float* duration, *float* index, *Type* updateType, *bool* countDown)**: The full Constructor which defines all fields for the Sine effect.
- **Sine(*float* divisor, *float* magnitude, *float* duration, *Type* updateType)**: An expedited Constructor which sets index to 0f, effectActive to false, and countdown to false.

Getters (updateType):

- **GetSine()**: Performs the operation Mathf.Sin on the current value of *index*.
- **GetCos()**: Performs the operation Mathf.Cos on the current value of *index*.
- **GetSineMagnitude()**: Performs the operation Mathf.Sin on the current value of *index* multiplied by *magnitude*.
- **GetCosMagnitude()**: Performs the operation Mathf.Cos on the current value of *index* multiplied by *magnitude*.
- **GetSineDuration()**: Returns a value of the Sin of *index* (multiplied by *magnitude*) relative to the overall *duration*. As the effect nears completion, this value approaches zero.
- **GetCosDuration()**: Returns a value of the Cos of *index* (multiplied by *magnitude*) relative to the overall *duration*. As the effect nears completion, this value approaches zero.

Utility Methods:

- **SetSineData(*SineData* sineData)**: A helper method which unpacks a SineData Scriptable Object and assigns it to the various fields of this class.
- **Update(out *float* floatRef, *Type* updateType)**: Updates the floatRef parameter based on the operation specified in updateType, returns the value of *effectActive* (true / false).
    - **floatRef**: A variable to be passed by reference which will only be set to 0f if the effectActive is not true
    - **updateType**: Controls which Getter operation will be passed on floatRef
    - **Update(out *float* floatRef)**: Updates the floatRef parameter based on the updateType stored within the instance of the Sine Class, returns the value of *effectActive* (true / false).
    - **Example**: Here is an example of the Update method used within the "PlayerDeadHandler.cs" Script provided in the Example folder.

```
void Update() {
//Use fadeOut's Update to manipulate the Spryt's Alpha
        float _alpha;
        if (fadeOut.Update(out _alpha)) //Update returns true only
when the effect is active, we only want to change the alpha if the
effect is active
            sSprite.Alpha = _alpha; //Only assign to the value of
alpha if the Effect is Active
    }
```

        - Since Update returns a true / false value, you can use it in the context of an if Statement so that the Spryt's *Alpha* is ONLY modified when "*effectActive*" is true.
        - The float "*_alpha*" holds the actual value of the Sine operation, which is what we want to assign to the Spryt's *Alpha* Property.
- **Increment()**: Increases *index* by the "slice of Pi" and handles the concept of duration. With no set *duration*, keeps *index* bound between 0 and 2*Pi.
- **Decrement()**: Decreases *index* by the "slice of Pi" and handles the concept of duration. With no set *duration*, keeps *index* bound between 0 and 2*Pi.
- **Activate(*bool* max)**: Sets *index* back to a value of zero and sets *durationElapsed* to false.
    - **max**: Maximizes the value of *index*, useful when counting down instead of up.
- **Activate()**: A zero-parameter alternative to Activate which sets *index* to zero.
- **Randomize()**: Used to set value to anywhere between a range of 0 to 2*Pi. Automatically sets *duration* to 0.