



# Spryt Lite Documentation

Spryt developed by [Bluish-Green Productions](#), for help, support, and feedback, please contact [Support](#)

## Table of Contents (Hyperlinked)

Features.....	2
Features in the Sample Scene.....	2
Getting Started.....	2
Compatibility: <i>Unity Version 2017.1.5f1 or later</i> .....	2
Setup.....	2
Importing Spryt Animator Pro.....	2
Importing Sprites into Unity.....	3
Sprite Setup.....	3
SprytLite Setup.....	4
FAQs (Frequently Asked Questions).....	5
SprytLite Scripting.....	6
Fields:.....	6
Properties:.....	6
Animation Methods:.....	6
Utility Methods:.....	6
Do MORE with Spryt Animator Pro!.....	7

# Features

Animate an instance with only a single script; No need for an Animation or an Animator!

- **One Component:** The quickest and easiest way to get animated Sprites into your project; Spryt Lite handles animation of varying speed (forwards or backwards) through a single Component without the need for any extra assets.
- **Runtime Control:** Easily modify the playback speed at runtime through Script!
- **Animation helper methods:** PlayOneShot, Pause, Resume, Toggle, Reverse

## Features in the Sample Scene

**Canvas:** The canvas to the left has numerous buttons which trigger the various animation helper methods available on an instance using SprytLite. While the speed slider in this Canvas has bounds, this is not the case when manipulating *speed* directly through Script. The “SpeedSlider” Script on the Slider Container acts as an in-between for the Spryt and the Slider.

# Getting Started

## Compatibility: *Unity Version 2017.1.5f1 or later*

SprytLite is 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).

Attempting to use SprytLite with an earlier version of Unity may function, but support is not provided for this.

## Setup

### 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.
2. **Import the Package into Unity:** With the project open in the Unity asset store, click the Import button.
3. **Import everything, or just SprytLite:** 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, but experienced users can exclude the Example folder. Once you have made your choice, select “Import”
4. **Errors? Don’t Panic:** If you are getting errors, please copy them and submit them to a new request for [Support](#).

## Importing Sprites into Unity

SprytLite makes use of Unity Sprites, so this documentation will provide a basic guide on importing your first sprites. 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.

## Sprite Setup

- **Sprite Renderer:** To add an animated instance to your project, select “**GameObject > 2D Object > Sprite**” (or simply right-click in the Hierarchy).
- **Assign a default Sprite:** 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 SprytLite will replace this default Sprite.
  - Also note that this step is not *strictly* necessary as SprytLite 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.

## SprytLite Setup

SprytLite 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 “SprytLite” Component to the GameObject. A quick way to do this is to type “ytl” into the text-field, unless you have other Scripts in your game which include this set of letters in their names.
  - **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 Sprite Renderer in the hierarchy.
- **Add Frames:** Now, you can simply drag-and-drop the frames for the animation directly onto the “Sprite Frames” property of SprytLite, 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.
  - **Re-use frames:** In certain circumstances, an animation may use the same frame multiple times throughout its duration. Since SprytLite deals with simple Lists of Sprites, this can easily be done by dropping the repeated frame(s) into the List wherever they are needed.
  - **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, SprytLite is ready to go, although you may wish to tweak the Speed property of the animation based on how many frames you are using.
- **Advanced Control:** You will need to create a Script to handle manipulating properties at runtime and calling the animation helper methods. This “controller” functionality can be appended to an existing class (like a character controller).
  - **SprytLite Reference:** In order to manipulate the Properties of SprytLite, you will need a reference to it. For instance:

```
public SprytLite spryt;
```

    - This line declares a reference to an instance of “SprytLite” named “spryt” which will likely be populated using the Inspector. You could also make the field private and use GetComponent in Start or Awake.
  - **Manipulating the SprytLite Component:** Once you have a reference to the SprytLite Component, you can manipulate its properties. Here is a simple example, disabling the Visible property of the Spryt:

```
spryt.Visible = false;
```
  - **In greater Depth:** For a practical example on how to manipulate the properties of a SprytLite Component, see the sample “*SpeedSlider*” script provided in the demo Scene.
    - For a complete look at the Properties and Methods available, see the section “*SprytLite Scripting*” below.

# FAQs (Frequently Asked Questions)

- **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*”
- **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!)

# SprytLite 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.

## 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 AssignFrames is called with a List of Sprites.
- **Visible:** Controls whether the Renderer is enabled or not. When this property is false, the Update method is not executed (the frame index is not updated).

## Animation Methods:

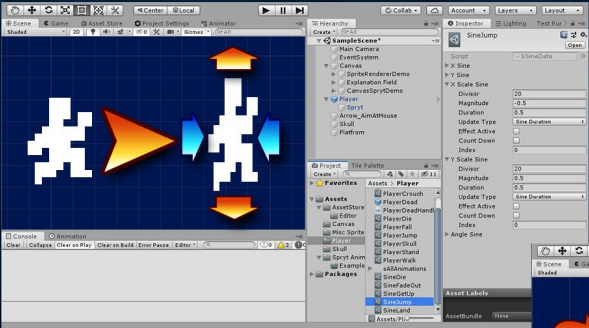
- **Pause():** Pauses the animation, freezing the Spryt on the current frame.
  - **Pause(int frame):** Pauses the animation on a specified frame.
- **Resume():** Resumes playing the animation and disables "playOneShot" behaviour.
- **Toggle():** Pauses the animation or Resumes it depending on its current 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?

## Utility Methods:

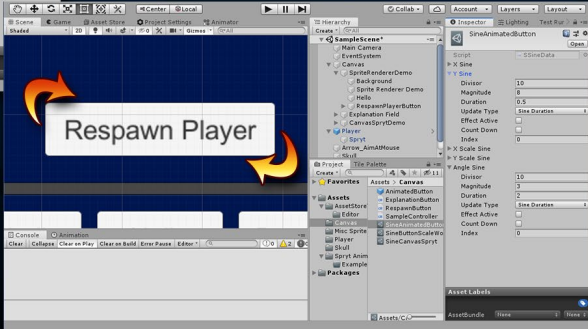
- **AssignFrames(List<Sprite> sprites):** A publicly accessible Method which can be used to assign a List of Sprites to the internal List used by Spryt.
  - **sprites:** A List of type Sprite, in the order the frames should be shown.
- **ResetSprite():** Resets all properties of the Spryt to default values as defined on the GameObject in the Inspector.
  - This will reset the speed, visibility, and frame index.

# Do MORE with Spryt Animator Pro!

Interested in additional functionality like *sprite-deformation*, an *Animation End Event*, built-in *multi-sprite* support? Check out [Spryt Animator Pro](#) on the Unity Asset Store.




**Create squishy Sprites and bouncy UI Images through integrated Sine Effects**




**Bring Static Images to Life**

# SPRYT



**Sprite Renderer Spryt**



**Canvas Image Spryt**

Swap Spryt Squash/Stretch Shudder

Visible Hide Show

Speed [Slider]

Playback Pause Resume Toggle Reverse

Alpha (0-1) [Slider]

White [Dropdown]

xScale [Slider]

yScale [Slider]

Angle [Slider]

Reset

**UI Compatible**  
Works on both *Sprite Renderer* and *UI Images*

# SPRYT

**Manipulate image properties at runtime without cumbersome Method calls or GetComponent calls**

**Without**

```

78 1 reference
79 public void HandleAnimations() {
80 //Handle Sprite scaling and speed
81 float hSpeed = Mathf.Abs(myBody.velocity.x);
82 float newScale = Mathf.Abs(transform.localScale.x) * Mathf.Sign(myBody.velocity.x);
83 if (hSpeed > 0.01f) {
84 //Set xScale to +/- depending on direction of movement
85 transform.localScale = new Vector2(newScale, transform.localScale.y);

```

**With Spryt**

```

78 1 reference
79 public void HandleAnimations() {
80 //Handle Sprite scaling and speed
81 float hSpeed = Mathf.Abs(myBody.velocity.x);
82 float newScale = Mathf.Abs(transform.localScale.x) * Mathf.Sign(myBody.velocity.x);
83 if (hSpeed > 0.01f) {
84 //Set xScale to +/- depending on direction of movement
85 mSprite.xScale = newScale;

```

**Without**

```

52 0 references
53 private void Start() {
54 //SpriteRenderer sprite = GetComponent<SpriteRenderer>();
55 sprite.color = new Color(sprite.color.r, sprite.color.g, sprite.color.b, 0.5f);
56 }
57
58 0 references
59 private void Move() {
60 }
61
62 0 references
63 void Update() {
64 }
65
66 0 references
67 public void Stealth() {
68 mSprite.Alpha = 0.5f;
69 }

```

**With Spryt**

```

52 0 references
53 private void Start() {
54 //SpriteRenderer sprite = GetComponent<SpriteRenderer>();
55 sprite.alpha = 0.5f;
56 }
57
58 0 references
59 private void Move() {
60 }
61
62 0 references
63 void Update() {
64 }
65
66 0 references
67 public void Stealth() {
68 }

```

**Simplified Syntax**

**SPRYT**

**Do away with unreliable String references when switching between animations**

```

21 0 references
22 void Start() {
23 }
24
25 0 references
26 void Update() {
27 Move();
28 HandleAnimations();
29 }
30
31 1 reference
32 public void HandleAnimations() {
33 bool isRunning = Mathf.Abs(myBody.velocity.x) > 0;
34 animator.SetBool("Running", isRunning);
35 }
36
37 0 references
38 private void OnCollisionEnter2D(Collision2D collision) {
39 }
40
41 1 reference
42 public void Move() {

```

```

25 0 references
26 void Update() {
27 Move();
28 HandleAnimations();
29 }
30
31 1 reference
32 public void HandleAnimations() {
33 bool isRunning = Mathf.Abs(myBody.velocity.x) > 0;
34 mSprite.Index = isRunning ? sWalk : sStand;
35 }
36
37 0 references
38 private void OnCollisionEnter2D(Collision2D collision) {
39 }
40
41 1 reference
42 public void Move() {

```

**No Strings**

**SPRYT**