

# Sprite

**View online:** <https://www.construct.net/en/make-games/manuals/construct-3/plugin-reference/sprite>

---

The **Sprite** object is an animatable image that appears in the project. It is one of the most important objects for most Construct projects. It is used to make most visual elements in a project, such as the player, enemies, projectiles, explosions and non-tiling scenery. (Tiled scenery is much better done with the [Tiled Background](#) object.)

If a Sprite has a single animation with a single frame, it just shows an image without animating. However, multiple animations can be added to Sprite objects with the [Animations editor](#).

All [instances](#) of Sprite objects share their animations. In other words, there is a single set of images comprising the animations which belongs to the [object type](#), and these images are referenced by instances.

Sprites can have effects applied. For more information, see [Effects](#).

## Scripting

When using JavaScript or TypeScript coding, the features of this object can be accessed via the [ISpriteInstance script interface](#).

## Sprite properties

---

### Animations

Click the *Edit* link to open the [Animations editor](#) for the object. All instances of the object type share a single set of animations.

---

### Size

Click the *Make 1:1* link to size the selection at original size (100%). This makes the width and height of the object the same as its first animation frame image.

---

### Initially visible

Set whether the object is shown (visible) or hidden (invisible) when the layout starts.

---

### Initial animation

Set the initially displaying animation.

---

### Initial frame

Set the initially displaying animation frame from the object's initial animation. This is a zero-based index, so the first frame is 0.

### Enable collisions

Enable or disable collisions for the object. Disabling collisions means no collision events will register for the object nor will any behaviors on the object register collisions with solids or jump-thrus. Disabling collisions does not improve performance unless there are some events or behaviors that test collisions.

### Preview Paid plans only

Enable to run a preview of the initial animation directly in the Layout View.

## Sprite conditions

For conditions in common to other objects, see [Common conditions](#).

### Compare frame

Compare the current animation frame number, which is a zero-based index (the first frame is 0).

### Compare frame tag

Compare the current animation frame tag.

### Compare speed

Compare the speed of the current animation, in animation frames per second. Animations which are playing backwards (e.g. with ping-pong animations) have a negative speed.

### Is flipped

### Is mirrored

True if the object has been flipped or mirrored with the *Set flipped* or *Set mirrored* actions.

### Is playing

True if a given animation is currently set. Animations are identified by their name (case insensitive).

### On any finished

Triggered when any animation reaches the end. Looping animations do not finish.

## On finished

Triggered when a given animation reaches the end. Looping animations do not finish. Animations are identified by their name (case insensitive).

## On frame changed

Triggered whenever the animation switches to another frame while the animation is playing.

## Collisions enabled

True if the object's collisions are currently enabled.

## On image URL loaded

### On image URL failed to load

Triggered when *Load image from URL* finishes downloading the image and is ready to display it, or if the load fails.

## Sprite actions

For actions common to other objects, see [Common actions](#).

### Add/remove animation

Add or remove an animation with the provided animation name. When adding a new animation, the animation name must be unique, and the new animation will have a single transparent frame sized 100x100. The last animation cannot be removed, as Sprite objects must have at least one animation.

*Animations are shared between all instances of the same object type, so this action will affect all instances regardless of which are picked in the conditions.*

### Add/remove animation frame

Add or remove an animation frame in the animation with the given name. When adding a new frame, it will be transparent and sized 100x100. The last animation frame cannot be removed, as an animation must have at least one frame. The *Where* parameter is the zero-based index, or animation frame tag, of the location to modify, and can be -1 to refer to the last frame in the animation. When adding an animation frame not at the end, it is inserted just before the given frame.

*Animations are shared between all instances of the same object type, so this action will affect all instances regardless of which are picked in the conditions.*

## Set animation

Change the currently playing animation to another animation. Animations are identified by their name (case insensitive). The new animation can either play from the *beginning* or from the same frame number as the last animation was on (*current frame*).

*Note that if the set animation is already playing, this action does nothing, even if set to play from the beginning. If you intend to restart the animation, use the Start action and choose from beginning.*

## Set flipped

Set whether the object image appears vertically flipped or normal. This also affects image points and the collision polygon. This is a shortcut for inverting the height (a flipped Sprite's height is a negative size).

## Set mirrored

Set whether the object image appears horizontally mirrored or normal. This is useful for platform games. Mirroring also affects image points and the collision polygon. This is a shortcut for inverting the width (a mirrored Sprite's width is a negative size).

## Set frame

Set the current animation frame that is showing, either by its zero-based index, or a string of the animation frame tag. If a tag is provided and there are multiple animation frames with the same tag, then it will use the first one. After this action the animation will continue to play at its current speed.

## Set repeat-to frame

Set the frame to return to when looping in the current animation, either by its zero-based index, or a string of the animation frame tag. This essentially changes the *Repeat to* property of the animation in the Animation Editor. It is especially useful when reversing animations, since the default of repeating to the first frame is no longer suitable. Instead when playing looping animations in reverse it is more useful to repeat to the last frame of the animation (so the animation actually repeats, instead of getting stuck on the first frame).

## Set speed

Set the playback rate of the current animation, in animation frames per second. Instances can have different animation speeds. You can also use negative speeds, which causes the animation to play backwards. Note in this case repeating animations should set the *Repeat to* frame at the *end* of the animation, otherwise by default it repeats to frame 0 (the start of the animation), causing the animation to stop after playing in reverse.

## Start

If the current animation is stopped, start playing the animation again. Playback can either resume from the *current frame*, or restart from the *beginning*.

## Stop

Stop the current animation from playing. The object will be left showing the current animation frame.

## Spawn another object

Create a new instance of a given object type. The new instance is created at the current object's position and also set to the same angle. The created object can be on any layer (chosen by its name or its zero-based number), and it can be positioned by an image point instead of the object's origin (chosen by its name or number). If a [Family](#) Paid plans only is created, a random object type in the family is picked. Tick *Create hierarchy* when creating the root object in a hierarchy to automatically create the rest of the scene graph hierarchy with connections in place. Choose a valid *Template name* so the new instance is created based on the template rather than an arbitrary instance.

See [Setting up a hierarchy in the Layout View manual entry](#) for more information about hierarchies.

When *Create hierarchy* is ticked, the additional objects created are also picked. This means subsequent actions for those objects will only affect the newly created ones.

When using this action with a family a *Template name* can be used, but will only take effect if the object type that is going to be created has that template name, otherwise it will be ignored.

See the [Templates manual entry](#) for more information on what templates are and how to start using them.

## Set scale

Sets the width and height to a multiple of the object's original size, similar to zooming the object proportionally. For example, if the object is 50x100, Set scale to 2 will set its size to 100x200, and Set scale to 0.1 will set its size to 5x10.

## Load image from URL

Load an image from a given URL. The current animation frame will be replaced with the image. It is not shown until the image has finished downloading, and *On image URL loaded* triggers. Images loaded from different domains are subject to the same cross-domain restrictions as AJAX requests - for more information see the section on cross-domain in the [AJAX](#) object. Data URIs can also be passed as an image, e.g. from a canvas snapshot or webcam image. The *Size* parameter sets whether the Sprite object will be set to the image size when it loads, or whether to keep its current size and stretch the image.

*Note that as all Sprite instances share the same set of animations and frames, loading an image will replace the image for all instances of the object type. If you want to dynamically load images for individual instances, try adding animation frames and loading a different image in to each frame.*

## Set collisions enabled

Enable or disable collisions for the object. Disabling collisions means no collision events will register for the object nor will any behaviors on the object register collisions with solids or jump-thrus. Disabling collisions does not improve performance unless there are some events or behaviors that test collisions.

## Set solid collision filter

Enable or disable collisions with the [Solid behavior](#) according to tags. Specify tags using a string of space-separated tag names. In *Inclusive* mode, collisions are only enabled with solids that match any of the given tags; if no tags are specified, collisions are disabled with all solids. In *Exclusive* mode, collisions are disabled with solids that match any of the given tags; if no tags are specified, collisions are enabled for all solids (the default).

# Sprite expressions

For expressions common to other objects, see [common expressions](#).

## AnimationFrame

The currently displaying zero-based animation frame number.

## AnimationFrameTag

The string tag of the currently displaying animation frame.

## AnimationFrameCount

The number of animation frames in the current animation.

## AnimationName

A string containing the name of the currently playing animation.

---

### **AnimationSpeed**

The current playback rate of the current animation, in animation frames per second. If the animation is playing backwards (e.g. ping-pong animations), the animation speed is negative.

---

### **OriginalAnimationSpeed**

The speed of the current animation as specified in the Animations Editor. This does not change if the animation speed is altered at runtime. It is useful for setting the animation speed to a multiplier of the original speed.

---

### **ImageWidth**

### **ImageHeight**

The original dimensions of the object (its current animation frame image size), in pixels. Since objects can be stretched at runtime causing the normal *Width* and *Height* expressions to return different values, these can be used to get the original size regardless of the stretched size.

---

### **ImagePointCount**

Return the number of image points on the currently displaying animation frame of the object.

*The count excludes the origin.*

---

### **ImagePointX(nameOrIndex)**

### **ImagePointY(nameOrIndex)**

Retrieve the position of an image point on the currently displaying animation frame of the object. You can pass either the zero-based index of the image point, or a string of its name.

*When using an index, the origin is excluded, so 0 means the first added image point.*

---

### **PolyPointCount**

Return the number of collision polygon points on the currently displaying animation frame of the object.

---

### **PolyPointXAt(index)**

### **PolyPointYAt(index)**

Retrieve the position of a collision polygon point on the currently displaying animation frame of the object, by its zero-based index.

*The first poly point is repeated again at the end (at the index PolyPointCount) since it makes it easier to iterate through each edge of the collision polygon.*