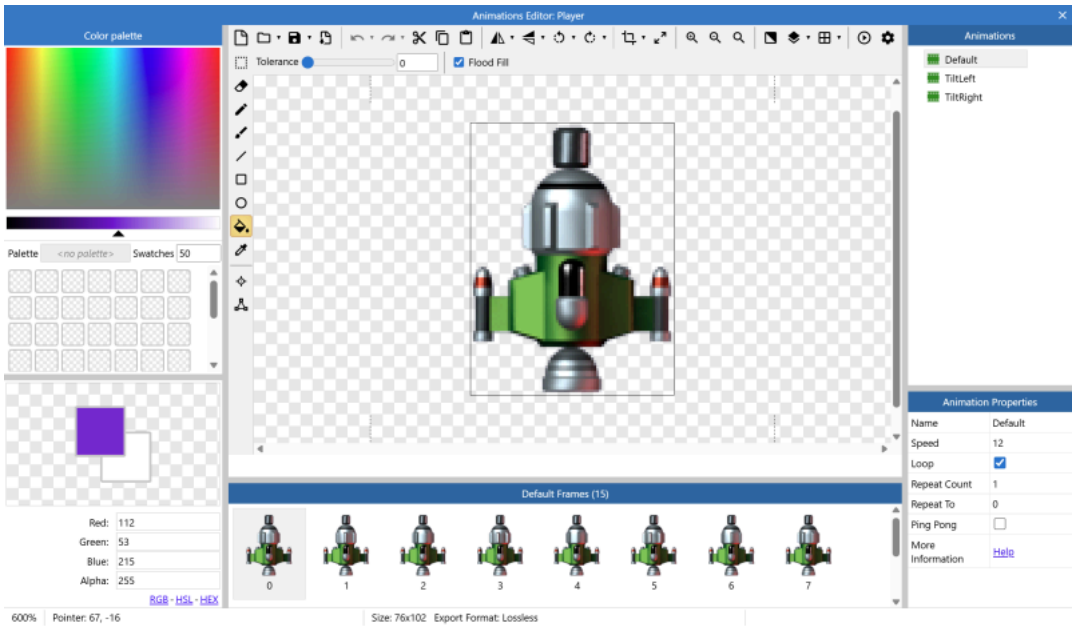
THE ANIMATIONS EDITOR

View online: https://www.construct.net/en/make-games/manuals/construct-3/interface/animations-editor

Construct has a fully-featured built-in image and animations editor, used to create animations for the Sprite object. When opening for an object without animations, such as Tiled Background, the animation editing features are hidden and it acts as a normal image editor. For brevity it is consistently referred to as the Animations Editor, even in cases where it is only editing a single image.

To open this editor, **double-click** an object with an image or animations in the Layout View or Project Bar.



The Animations Editor

Note each pane in the Animation Editor can be resized by dragging the borders, similar to how you can with the main Construct interface. This lets you customise the layout of the Animation Editor.

Color palette

The color palette appears on the left and allows a color to be picked for the drawing tools. You can choose both a primary and secondary color with left and right click. The pane also has a number of cells that can be used to remember a set of colors. Right-click a cell to save or use the primary or secondary color. By default left-clicking the cell will set the primary color.

Saving a palette

Right clicking anywhere in the area where swatches of the color palette are will bring up a context menu with the option Save palette, click it to save the palette and be able to retrieve it later. Saved palettes are associated with the object type or project file that is currently being edited, but can be used on other object types or project files.

Picking a palette

If the project has any saved palettes, one can be picked by either:

1. Clicking on the Pick palette option in the context menu.
2. Clicking the button on top of the swatches with the label **Palette**.

Any of those methods will bring up a dialog from which any of the palettes stored in the project can be picked.

Downloading palettes as a file

One of the secondary options of the Save tool, found in the top toolbar, is to save a single palette as a JSON file, or multiple palettes as a zip file. You can choose to save the currently selected palette, all the palettes available to the current object type or all the palettes available to the current project.

Loading palettes from a file

Similar to the Save tool, the Load tool has a secondary option to load palettes. It can load individual palettes or multiple ones bundled in a zip file. All the palettes are loaded into the current object type.

> *The tool only understands palette files in the format that is saved by the Save tool. If you wish to load a color palette generated by an external program, you will need to generate a file which the Animations editor can understand. See the color palette file format below for a description of it.*

Pasting colors

You can paste text specifying a color in to any of the color inputs to set the overall color. The text can be in any of the following formats:

- `r, g, b` or `r, g, b, a`
- `rgb(r, g, b)` or `rgba(r, g, b, a)`
- `hsl(h, s, l)` or `hsla(h, s, l, a)`
- Hex as either `#ffffff` or `ffffff`

Color components can be in the range **0% - 100%** or **0 - 255**. Alpha components should be between **0 and 1**.

Top toolbar: image tools

The top toolbar in the image pane provides tools that affect the entire image, such as mirroring and flipping. The tools available are as follows.

- **Clear:** resets the image to all transparent.
- **Open:** import an image from a local file. Note you can also choose **SVG** files, but these will be rastered in to a bitmap at a given size. The dropdown next to the main button also has an option to load color palettes from JSON or zip files.
- **Save:** export a copy of the current image. In the browser this downloads the current image as a PNG file. You can use the dropdown next to the button to save the current animation or all animations, bundled in a zip file. It is also possible to download the images with the associated image point and collision polygon data. One of the dropdown options allows you to save the current color palette to a JSON file, all the color palettes of the object type to a zip file or all the color palettes of the project to a zip file.
- **Set export format:** opens the *Image Format dialog*, allowing you choose whether the image is saved as lossless (i.e. perfect quality) or lossy (i.e. allowing some quality reduction in order to further reduce the file size) when the project is exported. The specific formats that these mean are chosen when exporting - for example lossless images can be exported as either PNG or WebP. This can also be applied to the current animation, or all animations. Note Construct stores all images in the project in a lossless PNG format; images are only converted on export.
- **Undo** and **Redo**: step through the change history.
- **Cut**, **Copy**, **Paste**: perform clipboard operations with the image.
- **Mirror** and **Flip**: invert the image on one of its axes. Use the dropdown next to the button to affect the entire animation.
- **Rotate anti-clockwise** and **Rotate clockwise**: rotate the image in 90°. Use the dropdown next to the button to affect the entire animation.
- **Crop:** resize the image smaller to remove transparent space around the edges of the image. This is a good idea to save memory. Note this leaves a 1px transparent border to improve the image quality at the edges. Use the dropdown next to the button to affect the entire animation or all the animations in the object type.
- **Resize:** resize the current image. A dialog will open with options for the resize, including a checkbox to apply the resize to the current animation or to all the animations in the object type.
- **Zoom in**, **Zoom out**, **Reset zoom**: adjust the zoom level in the image editor. Alternatively use **Control + mouse wheel**.

- **Toggle background brightness:** switch between a light and dark background for the image editor. Changing to a dark background can be useful when editing very light images.
- **Toggle onion skin:** Paid plans only display adjacent frames translucently over the current image when editing an animation. This can help when drawing animations. The options are to display the previous frame, next and previous frames, or next and previous two frames over the current image.
- **Grid:** toggle the display of a grid over the current image. Use the dropdown next to the button to adjust the grid settings, such as the grid size, color and whether to snap to the grid.
- **Preview:** preview the currently selected animation.
- **Settings:** open a dialog to adjust settings specifically for the Animations Editor.

Side toolbar: drawing tools

The side toolbar provides some tools for drawing in the image, as well as some extra Construct-specific tools for setting image points and adjusting the collision polygon. Some tools have extra settings, such as the size for the brush tool, which appear underneath the top toolbar. The following tools are available.

- **Rectangle select:** select, move, delete, cut, copy and paste rectangle sections of the image.
- **Pencil:** draw with a solid square, useful at 1px size for pixel art.
- **Brush:** draw with a soft round brush.
- **Line:** draw straight lines. Hold shift to lock the angle to 5° increments.
- **Rectangle:** draw a rectangle in the image. The secondary color is used as a border. Hold shift to draw a square.
- **Ellipse:** draw an ellipse in the image. The secondary color is used as a border. Hold shift to draw a circle.
- **Fill:** fill a continuous area of the image with a color.
- **Eye dropper:** select a color from the image. Alternatively hold Control and click with another tool selected.
- **Image points:** display and edit the origin and image points in the image. This switches the color palette pane to a list of image points, allowing you to add and remove image points. The origin determines the rotation point of the image, and is where the X and Y co-ordinates of the object are aligned to. Image points can be used in the event system to refer to alternative positions. For example you may place an image point at the end of a gun, so spawned bullets can appear at the end of the gun barrel instead of at the origin.
- **Collision polygon:** adjust the area that counts as colliding for this image. By default Construct guesses a collision shape, but it is not always accurate. Click and drag the points of the collision polygon to alter its shape. Right-click on a point or in a space to display a menu of additional options for the collision polygon, such as adding and deleting points. Some objects, like Tiled Background, do not use collision polygons.

Frames pane

The bottom pane displays a list of all frames in the current animation. Frames can be added and deleted here. Select a frame to switch to editing its image. Frames can also be **dragged and dropped** to adjust their sequence.

Selecting a frame shows a few properties in the properties pane:

- **Index:** the 0 based index of the currently selected frame, this property is for feedback only and can not be edited through the properties pane.
- **Duration:** is a multiplier for the amount of time to spend on the frame. For example, a frame duration of 2 will spend twice as long on that animation frame, 0.5 half as long, etc. relative to the current animation speed.
- **Tag:** a string to identify this animation frame at runtime.

Right-click a frame to duplicate or delete it.

Right-click in an empty space in the pane to see additional options for managing the animation, which include:

- **Add frame:** add a new empty frame at the end of the animation
- **Duplicate last:** add a new frame which is a copy of the last frame in the animation
- **Reverse frames:** invert the sequence of frames in the animation
- **Import frames**
  - **From files:** add multiple animation frames by selecting a set of local image files to import
  - **From sheet:** add multiple animation frames by selecting a local image file with multiple images placed on it (often called a *sprite strip*), and cutting it up in to individual images. A dialog shows up when selecting this option which allows to choose how the source image should be sliced and which images to extract from it.

You can also adjust the size of the frame icons appearing in the pane by adjusting the **Thumbnail size**.

Animations and properties panes

The right side of the Animations Editor shows the Animations pane, where animations can be added, edited and deleted. **Right click** a space to add a new animation or add a subfolder to organise animations. Right click an animation to see options like **Preview**, which shows how the animation will look in the game. You can also **Find all references** Paid plans only for an animation to locate all its references in events.

Right-click in an empty space in the animations pane to see a few context menu options, of note are:

- **Import animations**

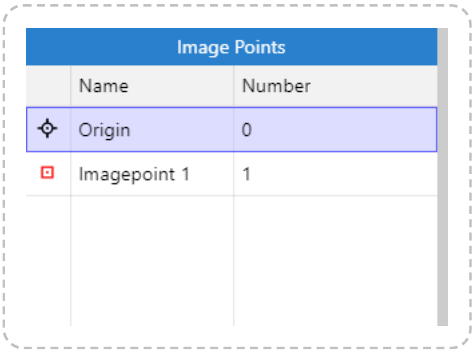
- **From files:** allows you to select multiple files to be added to a new animation. Also supports choosing zip files containing multiple images. In that case, a new animation is created for each zip file and each image found in the zip is added as a new frame of the corresponding animation. If the zip file contains a **c3-import-settings.json** file, it will affect the animation creation accordingly. See the **Bulk Importing** section for more information on that.
- **From sheet:** shows a dialog that allows to extract multiple animations from a local image file.

Selecting an animation also switches which frames are showing in the frames pane, and displays settings for the animation in the Properties pane. The following properties are available for animations.

- **Name:** the name of the animation. This can also be directly edited in the Animations pane.
- **Speed:** the rate at which to play the animation, in animation frames per second. For example if set to 5, each frame will last 1/5th of a second. Note this cannot be faster than the game framerate, which is typically 60. Set to **0** if you do not want the animation to play, which is useful if you want to control which frame is showing by events. 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.
- **Loop:** enable to infinitely repeat the animation.
- **Repeat count:** if the animation is not looping, the number of times to repeat the animation.
- **Repeat to:** the zero-based index of the animation frame to go back to when the animation loops or repeats.
- **Ping pong:** play the animation alternately forwards and backwards when looping or repeating.

Image points pane

When you select the **Image points** tool in the image editor, the left pane switches to a list of image points for the current animation frame.



The **Origin** is a special kind of image point defining the center of the object, or its point of rotation. It has a different icon to denote it. The term *image point* usually means "image points including the origin". Image points have a **zero based index**, and the first image point (number 0) is always the origin. The origin cannot be renamed.

You can also add additional image points. These are useful to create spawn points for other objects. Since you can create objects at image points in events, it is often useful to place an image point in places like the end of the player's gun in the image. Image points can also be given a name, and referred to in events by this name.

Editing image points

Select an image point in the list and a corresponding point will appear on the image. Left click to place the point under the mouse. The arrow keys can also nudge it 1 pixel in each direction.

An image point can be quickly placed using the **num pad**, e.g. **7** for the top-left corner or **5** for centered. Alternatively the image point can be **right clicked** in the Image Points pane and an option chosen from the **quick assign** menu.

Right clicking an image point in the Image Points pane also provides options to:

- **Apply to whole animation:** sets the image point in the same relative place in all frames in the current animation.
- **Apply to all animations:** sets the image point in the same relative place in all frames in all the animations of the object type.
- **Apply to all animations in subfolder:** sets the image point in the same relative place in all frames in all the animations in the same subfolder as the current animation.

If an image point does not exist in all frames, these options also creates it. **Holding shift** while placing the image point is a shortcut for **Apply to whole animation**.

Drag-and-drop

There are various ways you can import images by drag-and-drop in to the Animations Editor window.

Into main image pane

An image file can be dropped in to the main image editing pane to replace the content of the current frame with the dropped image file. This works the same way as using the **Open** button on the top toolbar.

Frames pane

Dropping a single image file in to the Frames pane will prompt you asking how the image should be treated. The image can be treated as a plain image file, in which case the image is added as a new frame in the current animation. This works the same way as the context menu option

Import▶From files. The image can also be treated as a sprite sheet, which works the same way as the context menu option Import▶From sheet.

Dropping multiple image files in to the Frames pane will add a new frame for each dropped image file. This works the same way as the context menu option Import▶From files.

Animations pane

You can drag-and-drop either a single image file or multiple image files in to the Animations pane, and it is handled in the same way as with the Frames pane, except that the frames are added to a new animation.

Bulk importing

The Animations Editor offers a few different methods to import images in bulk.

Importing folders

- Dragging and dropping a folder into the **Animations panel**, will create a new sub folder with a new animation inside of it. The animation will have frames for all images found at the root level of the folder. Both the new sub folder and the animation will have the name of the dropped folder.
- Dragging and dropping a folder into the **Frames panel** or the **Main drawing area** will add all the individual images in the folder as frames of the current animation.

Importing zip files

- Dragging and dropping a zip file into the **Animations panel**, will create a new animation with all images found at the root level of the zip file. The animation will have the same name as the zip file.
- Dragging and dropping a zip file into the **Frames panel** or the **Main drawing area** will add all the individual images in the zip file as frames of the current animation.

Importing from the toolbar options

The toolbar **Load** button has two options. **Load frames** and **Load animations**.

- **Load frames** allows you to pick images to be loaded as frames of the current animation. It is also possible to pick zip files from the file picker. All images in a zip file will be added as frames of the current animation.
- **Load animations** allows you to pick images to be loaded into a new animation. Is is also possible to pick zip files from the file picker. A new animation will be created with all of the individual images picked, while each picked zip file will correspond to a new animation being created.

Note: *loading from the toolbar does not support picking folders.*

Animated image file formats

In supported browsers, animated image file formats like GIF and APNG can be imported as a sequence of frames. This covers any method of importing an image file, including via drag-and-drop or by the toolbar *Load frames* option.

This is supported in Chrome and Edge. Other browsers may not support this feature, in which case only the first frame of the animation will be imported.

Nested content

Folders and zip files which in turn also have folders and zip files inside of them are supported, depending on how you try to import these, this is how C3 will behave.

- Dropping in the **Animations Panel:** C3 will process everything and create animations and sub folders as needed.
- Dropping in the **Frames Panel or Drawing Area:** all of the nested content will be turned into a flat list and added as frames to the current animation.
- Toolbar **Load Frames** option: all the nested content inside a zip file will be turned into a flat list and added to the current animation.
- Toolbar **Load Animations** option: if a zip file with nested content is loaded, the images at the root level of the zip will be used to create an animation, if other zip files are found, new animations will be created for them. If a folder is found in a zip file, a sub folder with an animation inside of it will be created, following the same patters as importing a folder.

The key points to remember are:

- Importing folders will create a sub folder with an animation inside of it, both named after the original folder. The animation will have frames corresponding to the images found at the root level of the original folder.
- Importing zip files will create an animation named after the zip file and will have frames corresponding to all the images found at the root level of the zip file.
- If nested content is found, the same pattern applies. Each folder will correspond to a new sub folder with a new animation inside of it. Each zip file will correspond to a new animation.

Note: *Construct does not allow duplicate animation names. If content with duplicate names is imported, Construct will always assign unique names to each animation created.*

Note: *By default Construct will assign unique names to imported sub folders aswell, but this is not strictly necessary and can be overridden using the "use-raw-folder-names" configuration option described below.*

Configuration file options

Normally when importing a single file, C3 asks if the file should be treated as a sprite sheet or as a single file. In the case of importing multiple files this isn't really an option. Because of that, when importing folders or zips, even if an animation ends up having only one frame, C3 never asks how it it should be treated.

To get around this problem, a special configuration file can be added to a folder or zip file to tell C3 how it should handle the files found on them.

It's a simple JSON file, must be named **c3-import-settings.json** and should look like this:

```
{
	"import-mode": "spritesheet",

	"sort": "alphabetical",

	"order": "ascending",

	"replace-existing-animation": false,

	"replace-existing-folder": false,

	"use-raw-folder-names": false,

	"spritesheet": {
		"horizontal-cells": 4,
		"vertical-cells": 4,
		"direction": "horizontal"
	},

	"svg": {
		"width": 100,
		"height": 100
	},

	"animation": {
		"name": "optional-animation-name",
		"speed": 5,
		"loop": false,
		"repeat-count": 1,
```

```
                "repeat-to": 0,
                "ping-pong": false,
                "frame-durations": [1, 2, 3, 4],
                "frame-collision-polys": [
                        {"points":[0,0,1,0,1,1,0,1]},
                        {"points":[0,0,1,0,1,1,0,1]},
                        {"points":[0,0,1,0,1,1,0,1]},
                        {"points":[0,0,1,0,1,1,0,1]}
                ],
                "frame-image-points":[
                        [{"originX":0.5,"originY":0.5}, {"name":"Image Point 1","x":0.5,"y":0.5}
                        [{"originX":0.5,"originY":0.5}],
                        [{"originX":0.5,"originY":0.5}],
                        [{"originX":0.5,"originY":0.5}]
                ],
                "frame-tags":["tag-1", "tag-2", "tag-3", "tag-4"]
        }
}
```

Configuration file options

- **import-mode** can be either "spritesheet" or "files".

  > ***Note:*** *The* ***import-mode*** *in the JSON file reflects the Animations Editor's interface, so it is only relevant when an animation with only 1 frame is going to be created.*

- **sort** can be either "alphabetical", "numerical" or "no-sort". Defaults to "alphabetical" if not provided.
  - **alphabetical** sort will interpret the file names as characters and will sort alphabetically.
  - **numerical** sort should be used when the names of the files are numbers, so the importer interprets the names as such and sorts as expected.
  - **no-sort** won't do any sorting and leave the incoming files in whatever order they were initially read in. This might not be the expected order.

  > *If a sort mode is not provided and all the files have numerical names* ***Ej. 1.png, 2.png, etc...****, then sorting will default to "numerical".*

- **order** can be either "ascending" or "descending". Defaults to "ascending" if not provided. Allows you to choose the sorting order of the imported files.
- **replace-existing-animation** is optional, can be either true or false and will default to false if not provided. If set to true, if an animation with the same name is found when importing, the imported one will replace the existing one, instead of creating a new one.

- **replace-existing-folder** is optional, can be either true or false and will default to false if not provided. If set to true, if there is an existing folder with the same name as the imported one, all the contents of the existing folder will be replaced with the content of the imported folder.
- **use-raw-folder-names** is optional, by default the importer will create unique folder names. Setting it to true will allow the importer to create folders with duplicate names. When using the Animations editor export with data options, the generated file has this option set to true.
- **spritesheet** mimics the settings you can pick when importing a sprite sheet through the Animations Editor UI.
- **svg** defines what size to use when encountering SVG files.
- **animation** these settings can be used to override the default properties of an animation. If an animation name is provided, then it will take precedence over the name of the folder or zip file.
- The **frame-durations** property of **animation** is an array that allows you to specify the frame duration of each frame in the animation. Add one number for each frame in the animation. If a frame does not have a matching value in the array, the default value of "1" will be used. If the array is not specified at all, all imported frames will have a duration of "1".
- **frame-collision-polys** is an array that allows you to specify the collision polygon for each frame in the animation. Add one object for each frame in the animation. If a frame does not have a matching object in the array, the default collision polygon will be used. If the array is not specified at all, all imported frames will have the default collision polygon. Each object in the array must follow the syntax in the example. This option is not meant to be used manually, instead it's values will be generated when using the options in the Animations editor to export animations with data.

  > ***Note: frame-collision-polys*** *can be used manually but is meant to be generated automatically by Construct when using the exporting options of the Animations editor.*

- **frame-image-points** is an array of arrays that allows you to specify the image points for each frame in the animation. The first element in each array always refers to the origin image point and must be specified using the syntax in the example, subsequent elements refer to image points and must also follow the syntax in the example. Add one array of image points for each frame in the animation. If a frame does not have a matching array of image points, the default image point will be. If the array is not specified at all, all imported frames will have the default image point. This option is not meant to be used manually, instead it's values will be generated when using the options in the Animations editor to export animations with data.

  > ***Note: frame-image-points*** *can be used manually but is meant to be generated automatically by Construct when using the exporting options of the Animations editor.*

- **frame-tags** is an array that allows you to specify the tag for each frame in the animation. If a frame does not have a matching tag in the array, the animation frame will default to having

an empty tag. If the array is not specified at all, all imported frames will have the default empty tag. This option is not meant to be used manually, instead it's values will be generated when using the options in the Animations editor to export animations with data.

> ***Note: frame-tags*** *can be used manually but is meant to be generated automatically by Construct when using the exporting options of the Animations editor.*

Sorting options examples

These are a few examples illustrating how the sorting options will behave when doing a bulk import.

Example 1

The following list of file names:

```
"pig.png", "octocus.png", "rock.png", "robot.png", "toaster.png", "monster.png"
```

when sorted in ***"alphabetical"*** and ***"ascending"*** order will end up like this

```
"monster.png", "octocus.png", "pig.png", "robot.png", "rock.png", "toaster.png"
```

Example 2

The following list of file names:

```
"1.png", "2.png", "3.png", "4.png", "5.png", "6.png", "7.png", "8.png", "9.png", "10.png"
```

when sorted in ***"alphabetical"*** and ***"ascending"*** order will end up like this

```
"1.png", "10.png", "2.png", "3.png", "4.png", "5.png", "6.png", "7.png", "8.png", "9.png"
```

> *Notice that in* ***Example 2****,* ***"alphabetical"*** *sorting does not work as expected with file names which are purely numbers.*

Example 3

The following list of file names:

```
"10.png", "9.png", "8.png", "7.png", "6.png", "5.png", "4.png", "3.png", "2.png", "1.png"
```

when sorted in ***"numerical"*** and ***"ascending"*** order will end up like this

```
"1.png", "2.png", "3.png", "4.png", "5.png", "6.png", "7.png", "8.png", "9.png", "10.png"
```

Example 4

The following list of file names:

```
"a-10.png", "a-9.png", "a-8.png", "a-7.png", "a-6.png", "a-5.png", "a-4.png", "a-3.png", "a-2.pn
```

when sorted in ***"numerical"*** and ***"ascending"*** order will end up like this

```
"a-10.png", "a-9.png", "a-8.png", "a-7.png", "a-6.png", "a-5.png", "a-4.png", "a-3.png", "a-2.pn
```

*Notice that in **Example 4**, **"numerical"** sorting does not work as expected because the file names are not purely numerical. In this case they can not be interpreted as numbers, even though they might look like they can, so no sorting is performed.*

In order to avoid unexpected behaviour, the best is to choose a naming convention for the files and stick to that, either numbers or characters.

It is worth mentioning that because of the way **"alphabetical"** sorting works it is possible to generate names that include both characters and numbers that will be sorted as expected.

Example 5

The following list of file names:

```
"anim-010.png", "anim-009.png", "anim-008.png", "anim-007.png", "anim-006.png", "anim-005.png",
```

when sorted in ***"alphabetical"*** and ***"ascending"*** order will end up like this

```
"anim-001.png", "anim-002.png", "anim-003.png", "anim-004.png", "anim-005.png", "anim-006.png",
```

*Notice that in **Example 5**, **"alphabetical"** sorting with numbers in the file names works as expected because the numbers are **zero padded**.*

Location of the configuration file

The configuration file can be placed at the root of a folder or zip structure and will affect all content found.

Another file with the same name can be placed further down the hierarchy and will take precedence over the ones found before. That way you can configure different sets of files to be interpreted differently when imported.

Exporting animations with data

The **Save** option allows you to export animations with their corresponding image point and collision polygon data. This can be done for an individual animation as well as for all the animations in the object type.

The generated file can then be used to bulk import all those animations into a different project. To do so, you can use any of the importing options described earlier.

Color palette file format

Bellow is an example of the file generated by C3 when it saves a color palette to a JSON file.

```
{
        "name": "MyPalette",
        "slots": "3",
        "palette": [
                "rgba(255, 0, 0, 1)",
                "rgba(0, 255, 0, 1)",
                "rgba(0, 0, 255, 1)"
        ],
}
```

Color palette file properties

- **name** is optional, if not provided the loaded palette will be given a unique name. If it is provided it will be used, unless it already exists in the project in which case it is used as a base for a unique name.
- **slots** is optional and represents the amount of swatches a palette has. If it is not provided the size of **palette** is used.
- **palette** is a JSON array of colors and must exist. The example uses the **CSS rgba** format for the colors, but any valid **CSS color format** can be used.

When downloading multiple palettes, C3 generates a zip file which bundles these type of files.

If you wish to import a color palette from an external program you will need to generate this type of file.

Miscellaneous settings

Clicking on the **Settings** button in the toolbar brings up a dialog where you can set the following options:

Import Settings

Guess collision polygon after importing images

Toggle this setting to choose if the collision polygon for new imported images should be guessed automatically or not. Affects all methods of importing.

Use file name when importing animations

Toggle this setting to choose if the file name of the imported file should be used to name a new animation or not. Affects all methods of importing animations.

Palette Settings

Swatch size

Choose the size the swatches of the color palette should have.

Download Settings

Image format

Choose the file format, quality and compression that the Animations editor should use when saving files to the local file system.