Timpart Spriter Test Suite

Version 0.2

6th December 2012

Table of Contents

Introduction

This collection of animations go through the features of Spriter's SCML file format one feature at a time. Plugin authors can use the test suite to gain additional confidence that their implementation is behaving correctly. BrashMonkey's SCML specification has many powerful features, which can seem overwhelming as a whole. The author hopes this test suite will help break down the specification into manageable chunks and allow plugin authors to feel they are making concrete progress with their implementations.

Descriptions are given for each test of how it should look and what to watch out for. Another way to see how an animation should look is to load it into Spriter then run it. In many cases there are extra animations available, with Fail in the name which simulate some typical problems that may be encountered. The Fail animations are not intended to be used as tests with a plugin – they have no hope of working correctly!

Each test has been individually crafted, and in some cases manipulated outside of Spriter to obtain the exact test condition wanted. Although you can load them into Spriter, never save one of the tests from Spriter, as Spriter may have cleaned up or optimised things and the saved version may differ in some important aspect from the original.

The author can be contacted via BrashMonkey's Spriter forums for comments and suggestions about this test suite. He regrets he is unable to help with advice on implementing SCML or debugging software. To avoid confusion do not distribute modified versions of this test suite. Please obtain only directly the source given in the BrashMonkey Spriter forums. At the time of writing this is TimpartSpriterTestSuite on DropBox. Each version has its own directory.

This test suite is expected to evolve with time and is at the moment limited in scope. To avoid confusion if you mention in your literature that your plugin passes tests in this suite you must mention which version of the suite was used (see first page). You should also state which tests were passed and which failed. You can refer to individual sections within the suite or the whole suite if the results were the same. E.g. The plugin passes all of Timpart Spriter Test Suite version 1.5 except for the Pixel Art Mode section.

Test Environment

The tests were prepared in Spriter's standard window without zooming. The test environment drawing area should be at least 750 pixels wide and 500 high, or be capable of scaling down from such dimensions. The origin of the animation should be placed near the centre of the drawing area. While the tests were being developed a white background was used (and the sample pictures are shown on such a background in this document). A light grey background should work as well though in some cases a few objects may not contrast well with it. Some black sprites are used, so unless you are willing to recolour them, a black background would not be suitable.

Some tests may involve around 130 sprites of 64 by 64 pixels being held in memory at the same time.

The animations in most cases last for many seconds, so it should be possible to look for the important aspects while they run at normal speed.

Legal Matters

The author has produced this Work as a private individual and it is not associated with his employer or BrashMonkey.

“Spriter”, “Spriter Character Markup Language” and “SCML” are the property of BrashMonkey and no infringement of their intellectual property is intended. Trade marks are acknowledged.

A test suite cannot prove a software product to be free of defects and no one must place any reliance on a software product passing the tests in this Work as any indication it is free of defects or fit for any purpose nor may they represent a software product as such.

This Work does not claim to give a complete or accurate diagnosis of faults in software products. A particular software product may have a fault mentioned but not exhibit the behaviour mentioned, or may have a fault not mentioned.

Users of this Work must not state or suggest the the author of the test suite endorses a software product.

Unless required by applicable law or agreed to in writing, the author provides the Work on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with your usage of the Work.

Any Product produced by a third party and tested (by any person) using this Work is the sole responsibility of that third party. The author does not provide the Product with warranties of any kind or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Product and assume any risks associated with your usage of the Product.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the author be liable to you for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the author has been advised of the possibility of such damages.

Versions

0.2 6th December 2012

Six more Basic tests, first two Rendering tests.

0.11 18th November 2012

No new tests. Added id attribute to <object/> in SCML which had been omitted by generator.

0.1 16th November 2012

Initial two Basic tests: Position and Rotation

The Tests

The tests are divided up into sections:

Basic

Rendering

Basic

The Basic tests are concerned with putting sprites on display, moving them without tweening and overriding their attributes. Later sections use features tested in the Basic section, so it is worthwhile making sure that an implementation passes these tests.

Most sprites used in this section have dimensions that are a power of two, to make it easier for developers starting on platforms that only readily support such sprites. In later sections the sprites can have any dimensions.

1) Position
2) Rotation
3) Scaling
4) Scaling and Rotation
5) Swap sprite
6) Looping to start
7) Z Order
8) File names

Rendering

The rendering tests check applying sprites on top of each other, including the action of transparency both of the sprite itself and imposing extra alpha transparency.

1. Transparent Sprites, Alpha Blending
2. Opacity Override, Alpha Blending

Basic

1. Position

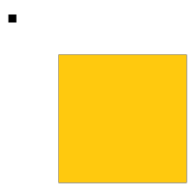
This test checks that x and y coordinates are correctly used and that an object can be positioned with both positive and negative coordinates relative to the origin. It also checks for correct usage of the pivot point which is used to describe what place should be positioned at the x,y coordinates. No rotation takes place in this test.

Expected Result

A square moves along a square path around the origin in a clockwise direction. The origin is marked by a small dot. The large square should start and finish diagonally above and to the right of the origin with two opposite corners on a line passing through the origin.

Check that the square moves down, then left, then up, then right back to the original position where it remains for half a second.. There are brief changes of pivot point in the middle of the bottom, left and top sides of the path. These do not affect the smooth movement of the square if pivot points are used correctly.

0 ms



1000ms

2000ms

3000ms

4000 - 4500ms

Unexpected Results

If the first move of the square is up rather than down then y axis is probably reversed and the square may start like this:

Y-axis reversed 0ms

The SCML coordinate system has increasing y values at the top of the display area. If you have this problem it is strongly recommended that you resolve it before diagnosing other problems.

If the big square is too far above the origin, and makes sudden erratic movements left and/or down half way along each side except the right hand one then perhaps the sprite is being positioned so that bottom left corner is at stated position all the time. It should be positioned so that the point identified by multiplying the pivot_x and pivot_y ratio by the side lengths of the sprite is positioned at x y:

Origin of sprite bottom left 0ms

If the square seems relatively unaffected but has a V shaped dip on the lower and upper part of the path, it is possible that your software has got the y-pivot point inverted. Subtracting it from 1 might help. Start position similar to that above

Notes

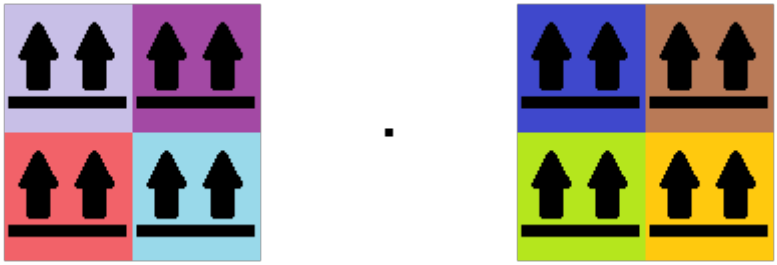
The movement is untweened and animated at twenty frames per second. The animation does not loop.

2. Rotation

This animation does not actually move, instead it has a single frame containing eight large squares, and a small square indicating the origin. Various pivot positions and rotations are used.

Expected Result

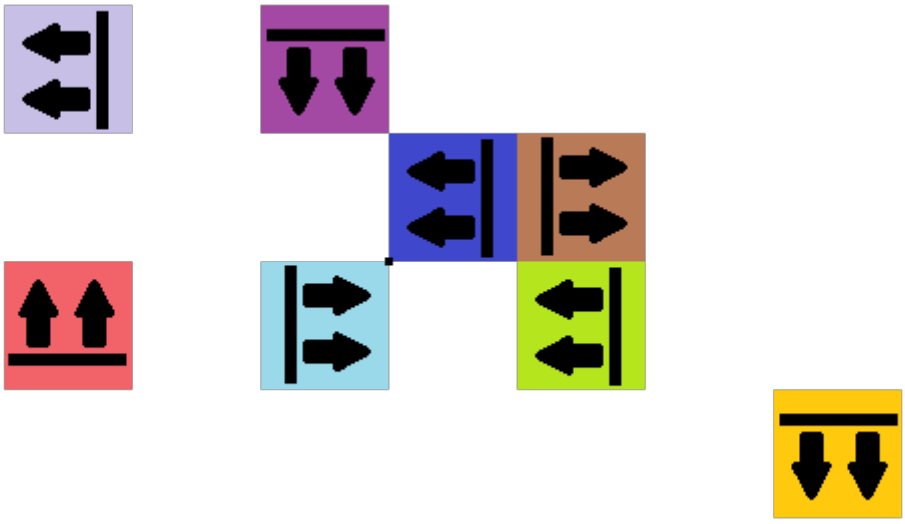
The squares have been rotated in various ways so that they are arranged to form two larger squares. Each individual square has a “This way up” symbol in it which should all point upwards.



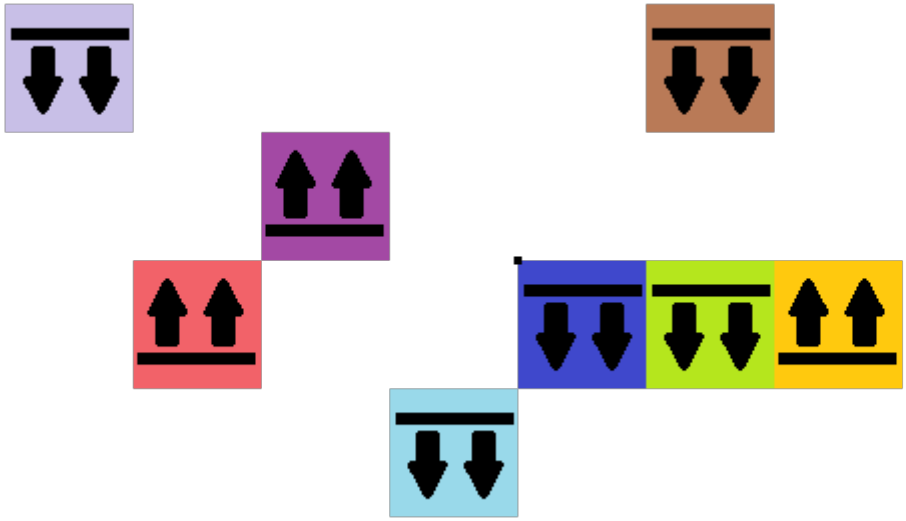
Unexpected Results

The following diagrams may help work out what has gone wrong. The first diagram shows the positions of the squares before rotation. The pink square, bottom left of the left hand group, does not move at all. The green square, bottom left of the right hand group, should only rotate about its centre. The other squares rotate about one of their corners, with the left hand group all using the bottom left corner, and the right hand group the other three possibilities.

If no rotation takes place the squares remain in their original orientations:



If the rotations are made clockwise, instead of anti-clockwise this is the result:



Notes

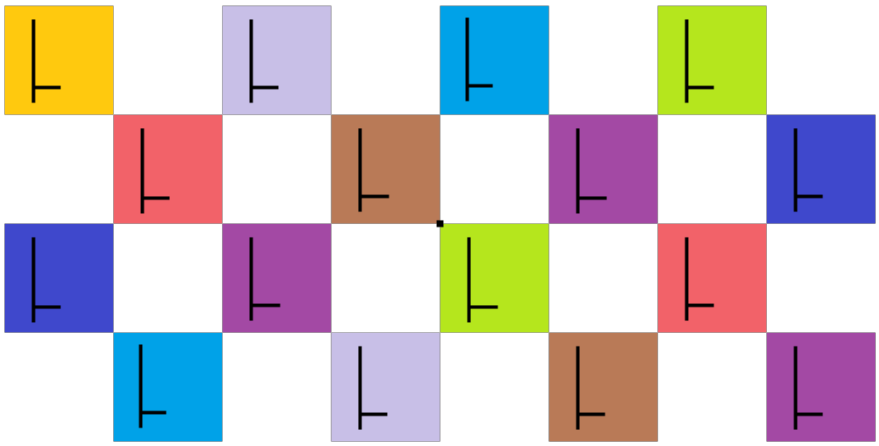
All the pivot points are at a corner of or within the sprite. A later test includes pivot points outside the sprite.

3. Scaling

This test checks scaling of sprites in both the x and y directions. A variety of rectangles are used, and each is scaled to the same 64 × 64 size. There is a symbol drawn in each square, which can be used to check for correct mirroring. No rotations are used in this test. The scale factors include 0.5, 2 and -1 in various combinations. When correctly scaled the squares form a chequered pattern. The top left square (orange) is not scaled at all and serves as a reference. The origin is marked with a small black square for convenience.
The top two rows all use 0,0 on the sprite to do the scaling from. The bottom two rows use other corners or the centre and in some cases -1 scale factors which result in mirroring.

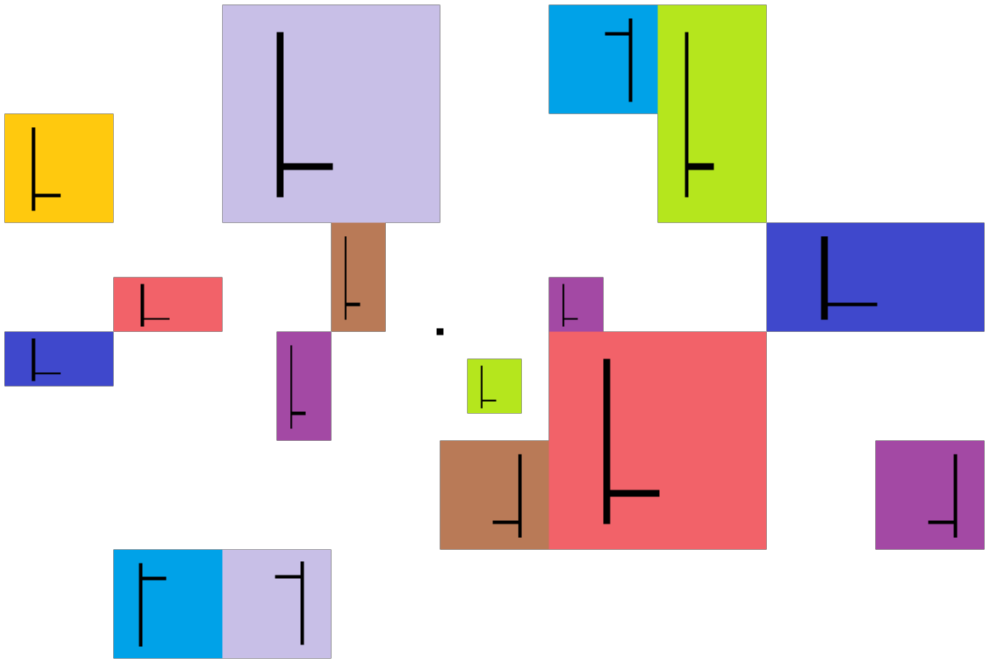
Expected Result

The squares are all the same size, and the symbols all face the same way (especially those on the bottom row). The origin is indicated by a small black square.

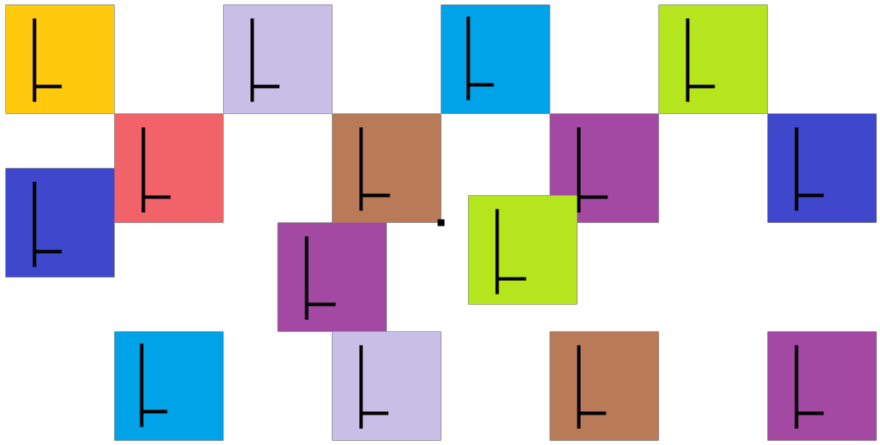


Unexpected Results

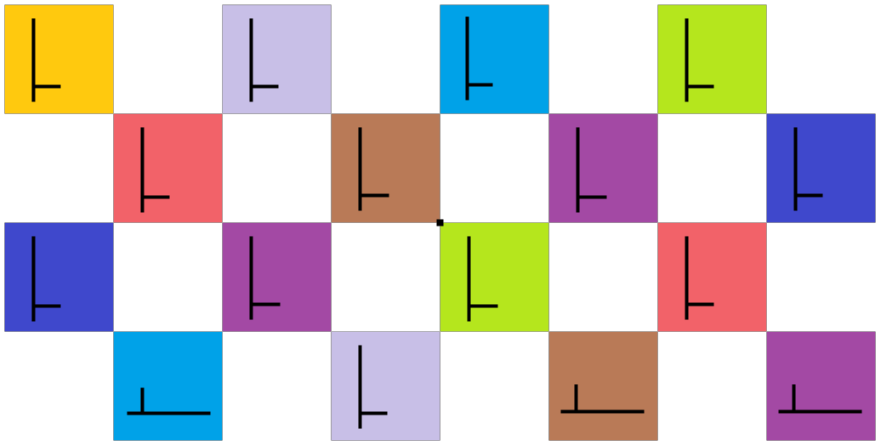
If no scaling takes place (or is ignored) then this is the result:



If all scaling takes place determining the pivot point of the sprites before the scaling rather than after as it should be then this may result:



If negative scale factors are being converted into rotations rather than the mirroring they should be then this could result (note the symbols in the bottom row):



4. Scaling and Rotation

This test combines scaling and rotation at the same time. Additionally some of the pivot points are outside the body of the sprite. (All pivot points in Basic 2 were at a corner of or within the sprite.

Expected result

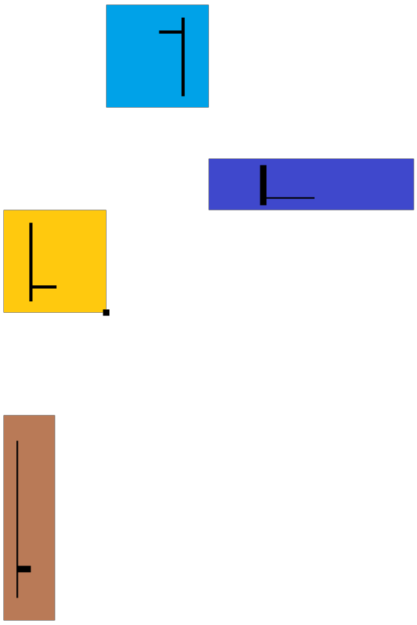
The sprites form a neat square. A small black square indicates the origin. As in Basic 3 the orange square has no rotation or scaling.

As two transforms take place, these diagrams may help understand what is happening. The original state is:

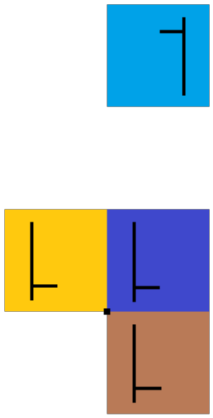
After (just) scaling it would look like this:

Unexpected Results

If the rotation is done before the scaling, this could result:



If the rotation goes the wrong way for the light blue square which is mirrored due to a negative scale factor, it would look like this:



Notes

This”animation” does not move. It runs for 5 seconds and doesn't loop.

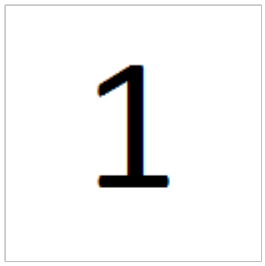
5. Swap

This animation confirms that sprites can be added and removed in an animation.

Expected Results

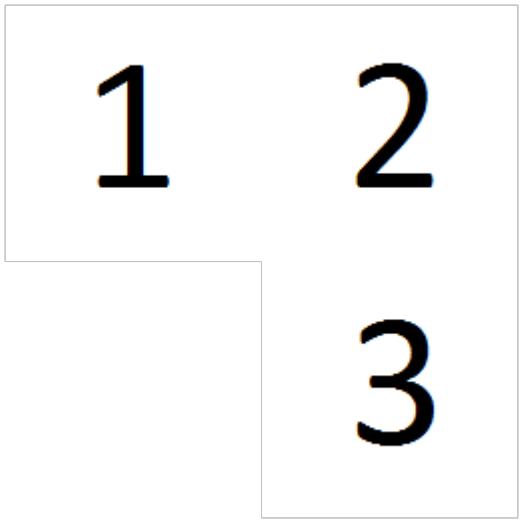
The animation starts and finishes with a frame with nothing in it. At half second intervals sprites with the numbers 1 to 4 are added individually then removed in same order they were added. The sprites do not change position.

0ms, no sprites at all

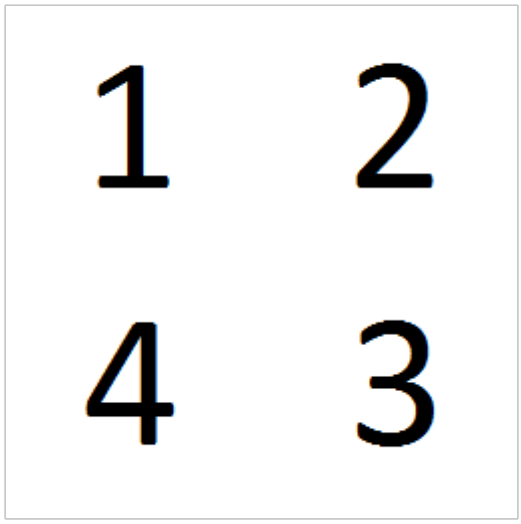


500ms

1000ms



1500ms



2000ms

2500ms

3000ms

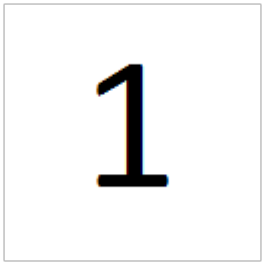
3500ms

4000ms empty again

Unexpected results

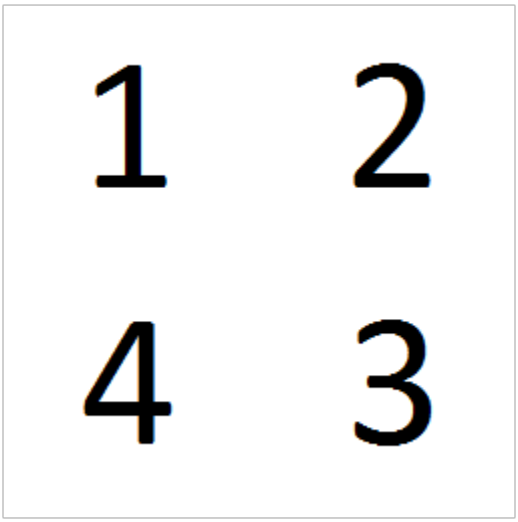
Problems are most likely to show up either at the start, or after the digits 1 to 4 are meant to disappear in turn starting at 2500ms.

If the implementation is wanting an object in each frame it might accidentally create one with all values defaulted. The first frame would have a digit 1 in it which would then move up and to the left at the next frame.



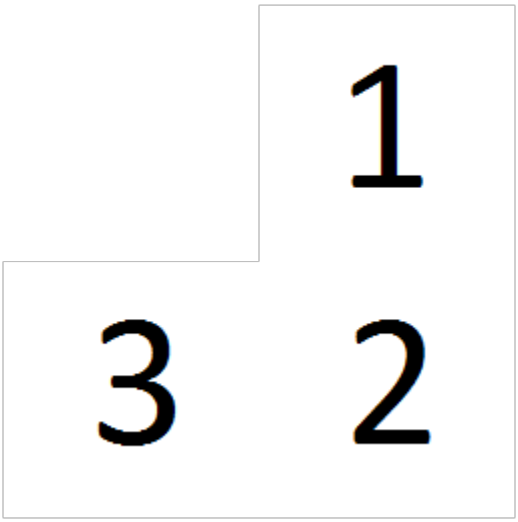
0ms defaulted object

If there is a problem with removing objects then the all four square display at 2000ms would persist.



2500ms

If the file attribute in the SCML is being ignored and the id or position in the file is being used instead then from 2500ms onwards the lower numbers are shown instead of the higher ones:



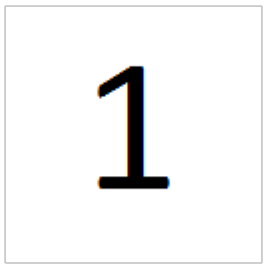
2500ms

6. Looping to Start

Up until now all animations have played just once. This animation loops, restarting at the default position of the beginning.

Expected Result

Numbers are displayed in the sequence 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 indefinitely.



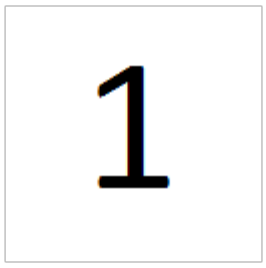
0ms

1000ms

2000ms

3000ms

4000ms



5000ms (start of repeat)

Unexpected Results

If the animation goes from 1 to 5 just once then the looping attribute of the animation isn't working properly.

If the animation goes 1 2 3 4 5 4 3 2 1 2 3 4 5 4 3 2 1 repeatedly then it is ping-ponging

Note

No Fail animations has been provided.

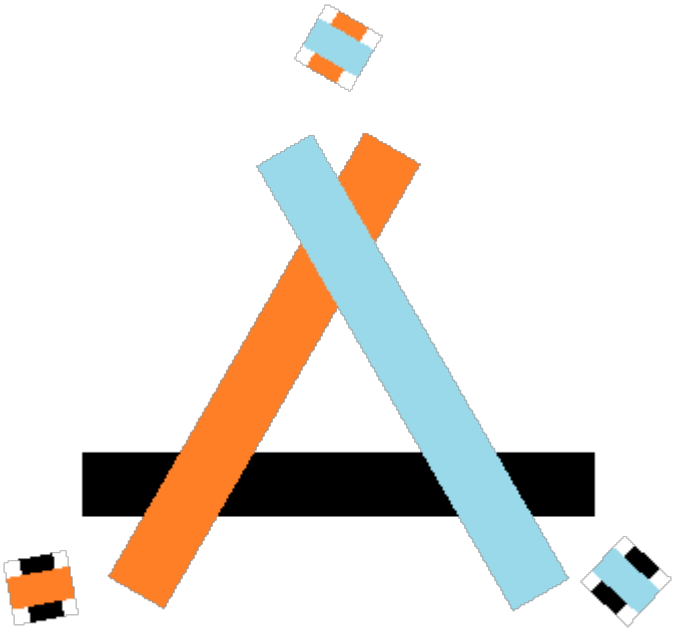
7. Z Order

This animation tests changes to the order in which sprites cover each other. Three rectangular sprites overlapping in a triangle are sequenced through all six possible combinations. At each corner a small indication of the correct overlap has been placed.

Expected Result

The animation goes through a sequence of six arrangements, changing every three seconds.

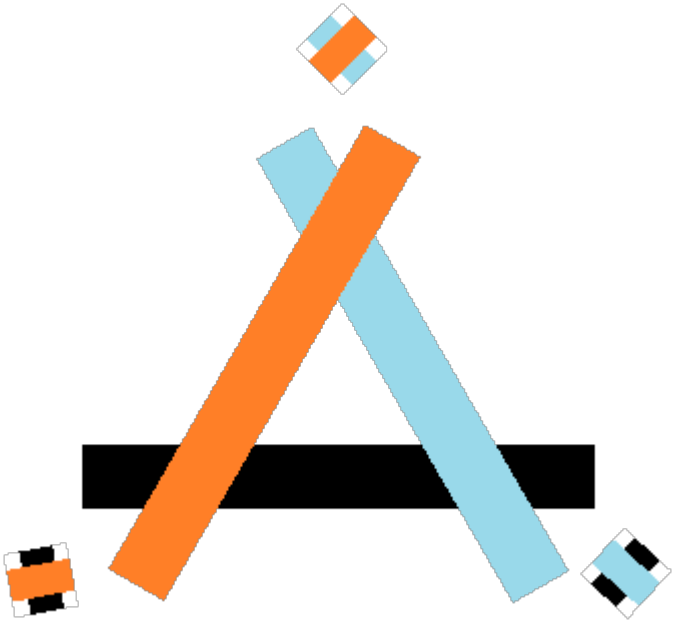
The start position is



0ms

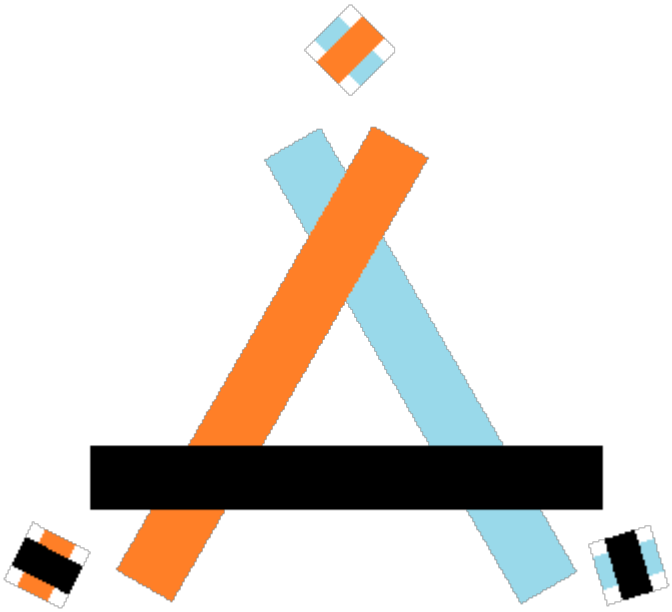
Check each overlap is correct by comparing it with the nearby small X arrangement of two colours.

After three seconds the orange bar moves on top:



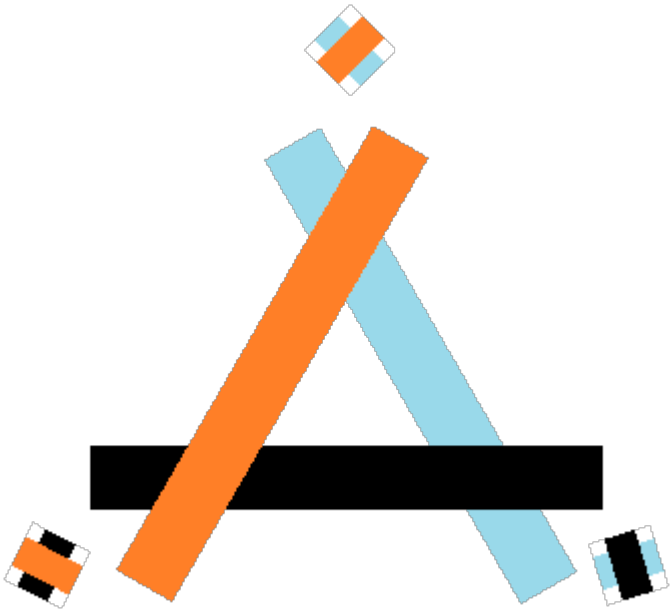
3000ms

Then the black bar goes on top:



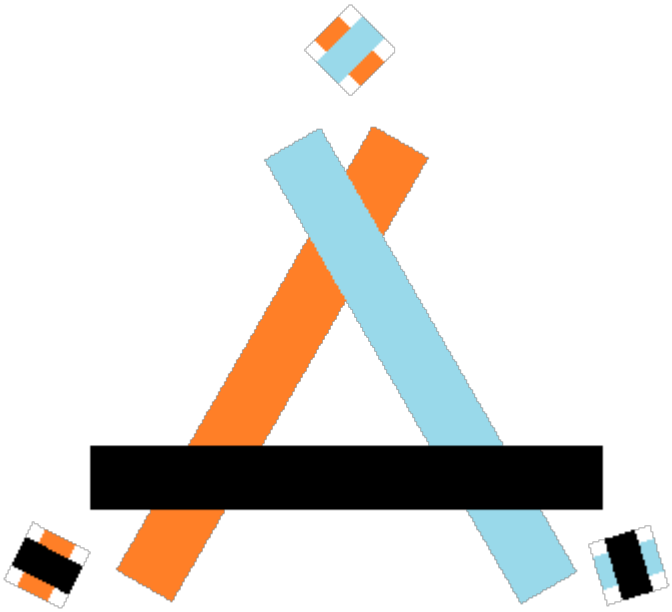
6000ms

Next the orange bar goes on top:



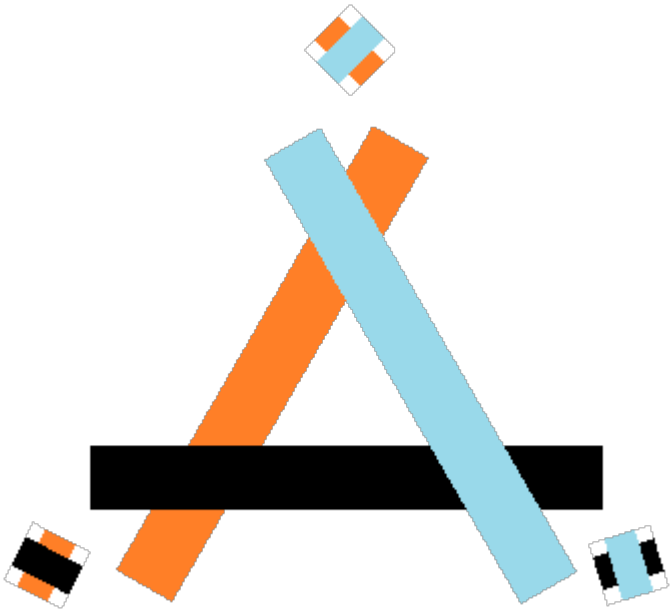
9000ms

Followed by black going to the top and orange to the bottom:



12000ms

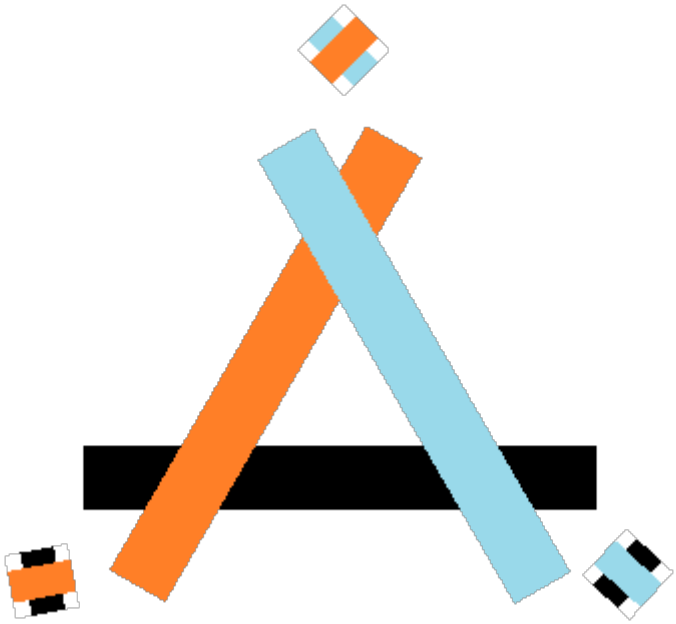
Finally blue goes on top:



15000ms

Unexpected Results

If the configuration stays in the first arrangement so that in later frames they mismatch with the indicator X's, then perhaps the order in which the sprites have been defined is being used, rather than the order they appear in an individual frame. This is what the *second* frame would look like; note the top crossing mismatches:



3000ms

If the Z-Order is being reversed then every crossing would mismatch with the indicator. Here is what the first frame would look like:

0ms

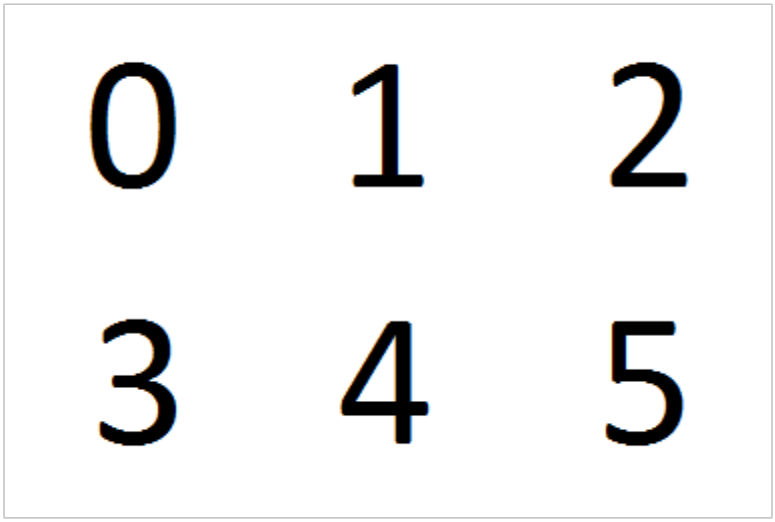
8. File Names

This static “animation” tests that sprites can be loaded from the same directory as the SCML file as well as when within nested subdirectories.

Expected Result

Six squares numbered 0 to 5 should be shown. 0 comes from the same level as the SCML file, 1 is in a directory, 2 to 5 are in nested sub-directories

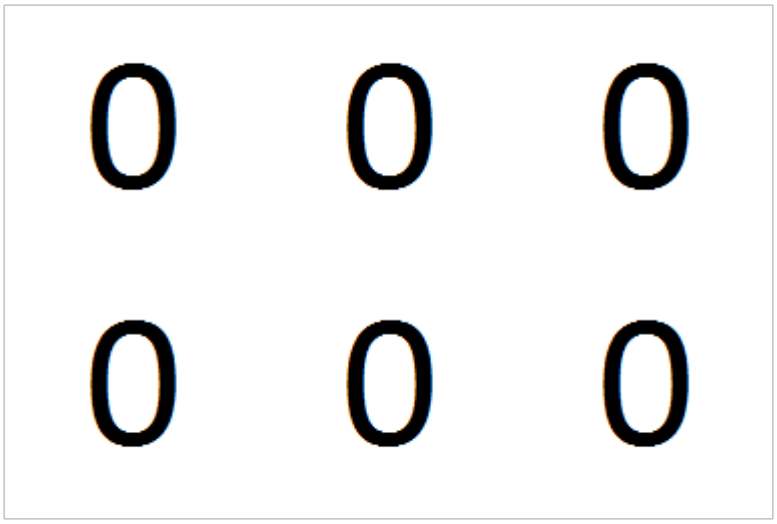
Currently some Spriter alpha releases are unable to load the SCML file properly and does not display the correct result.



Unexpected Results

If sprites in subdirectories cannot be loaded then this may be the result:

If the SCML folder attribute is being ignored, everything would be displayed from folder 0 and the result would be:



Rendering

1. Transparent Sprites, Alpha Blending

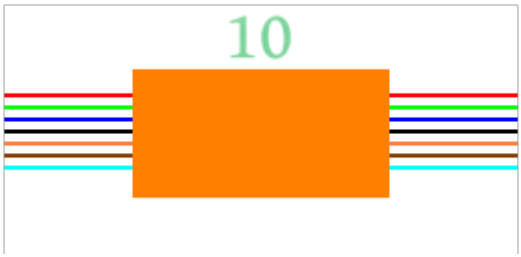
This animation shows a sequence of squares with increasing transparency, alternating between orange and blue. The squares are shown on top of a white background with horizontal coloured stripes.

Each square is shown to the right of a reference square showing the correct appearance, forming a rectangle. Although the left hand squares have the appearance of being transparent, they are in fact fully opaque screen captures of Spriter showing the right hand square, and carefully lined up.

Expected Result

A sequence of sprites with increasing transparency are shown at half second intervals, going from completely opaque to completely transparent. Alternating orange and blue sets are shown. Small numbers count down at one second intervals to help keep track.

The two halves of the coloured rectangle (composed of two adjacent squares) should match. The left side is the correct appearance.

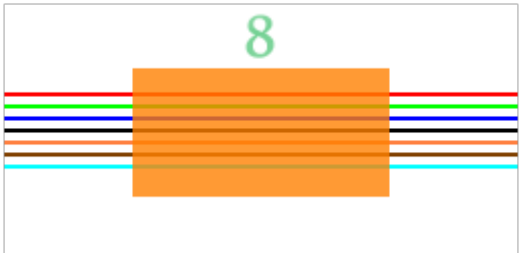


0ms

500ms

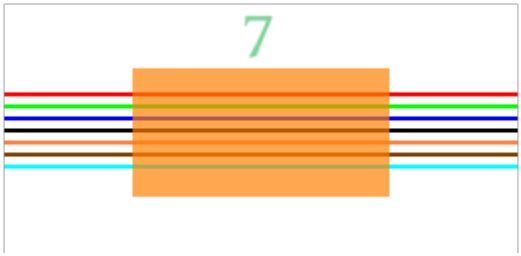
1000ms

1500ms

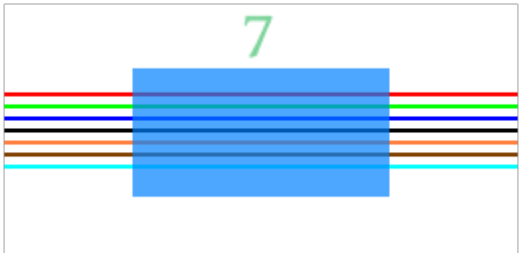


2000ms

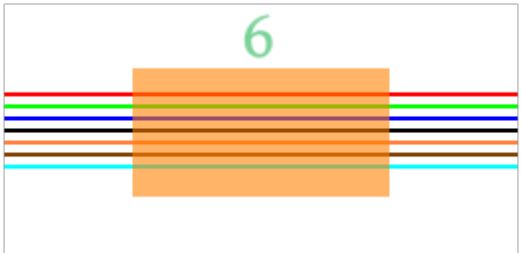
2500ms



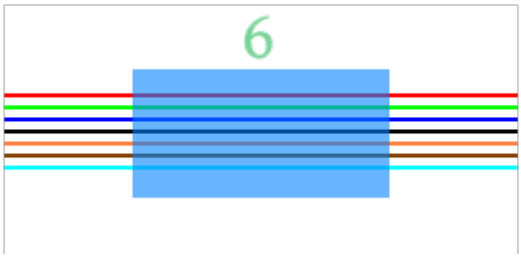
3000ms



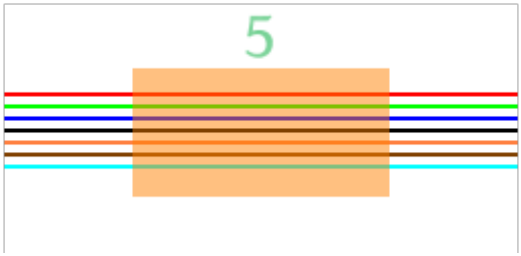
3500ms



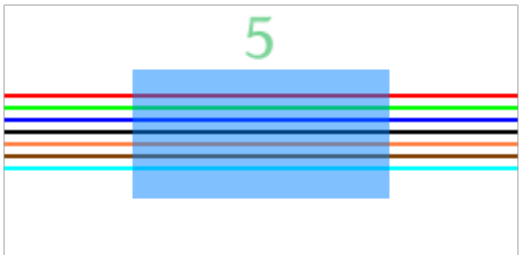
4000ms



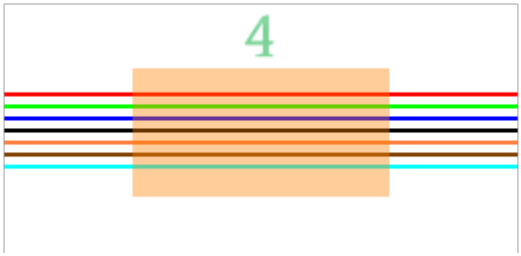
4500ms



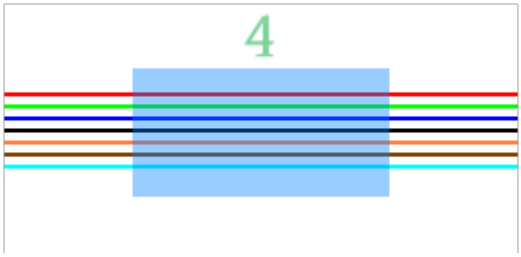
5000ms



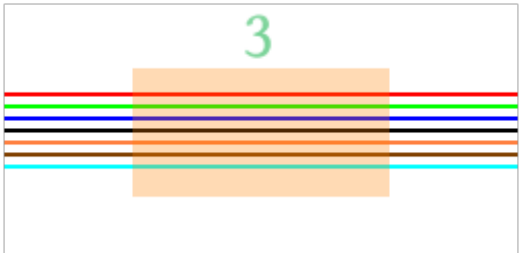
5500ms



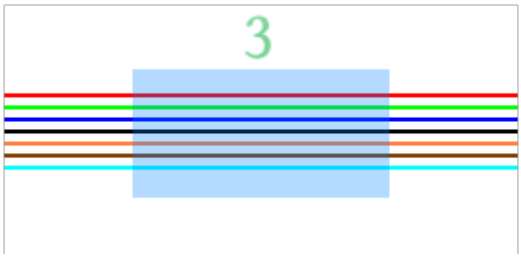
6000ms



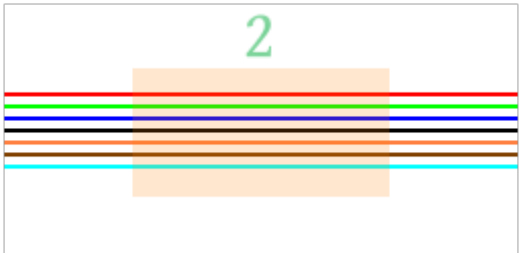
6500ms



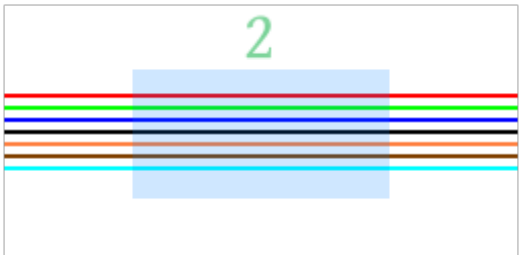
7000ms



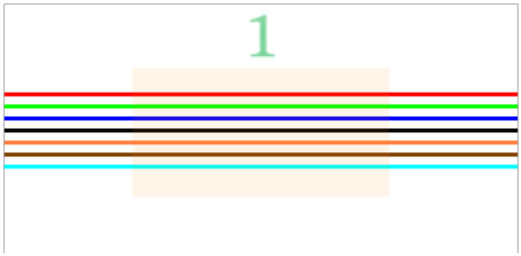
7500ms



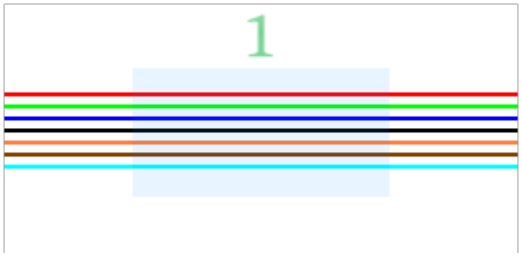
8000ms



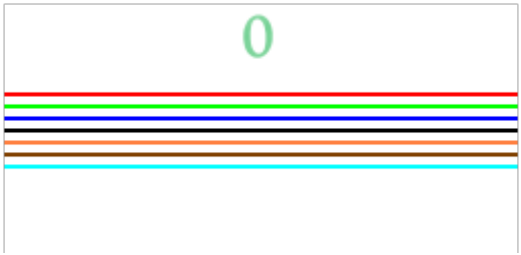
8500ms



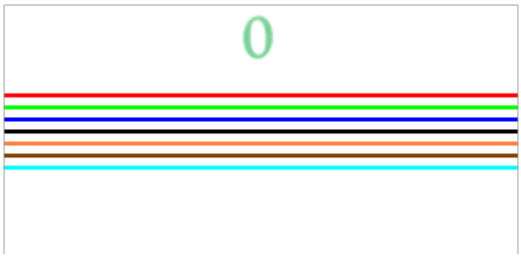
9000ms



9500ms



10000ms

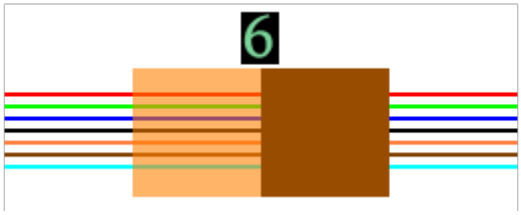


10500ms

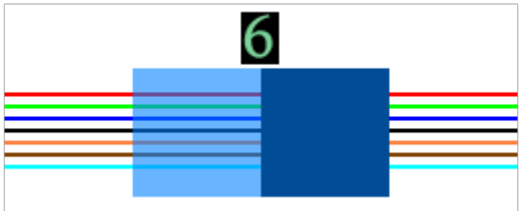
Unexpected Results

If the transparent sprite is alpha blended in a separate area and then overwritten onto final display then one of the following might result. Note that stripes are not visible under the right hand square.

With black:

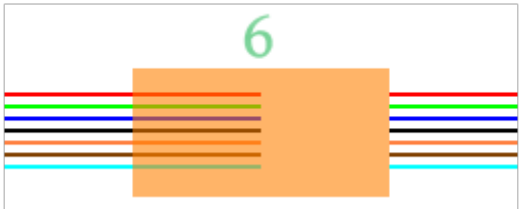


4000ms

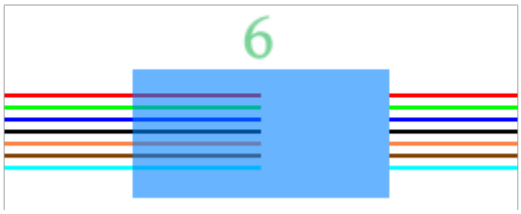


4500ms

With white:



4000ms



4500ms

Notes

The screen shots were captured using a version of Spriter running on Windows. It is expected that other operating systems / graphics cards would give the same result.

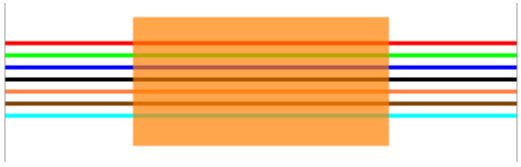
2. Opacity Override, Alpha Blending

Sprite allows the transparency of a sprite to be overridden. This is the opacity setting in Spriter and the a (for alpha) attribute in SCML.

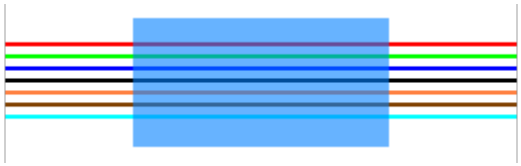
This animation shows a sequence of four squares in a similar manner to the previous test. The first two are opaque, the last two are already slightly transparent and are made even more so.

Expected Result

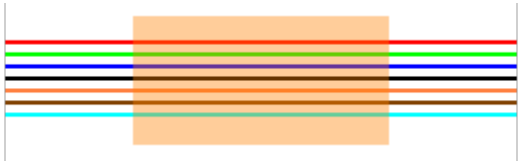
The sequence should appear as follows. As in the previous test the left hand side of the rectangle is a square showing the correct appearance.



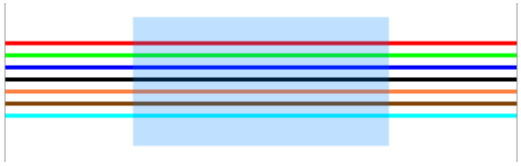
0ms



1500ms



3000ms

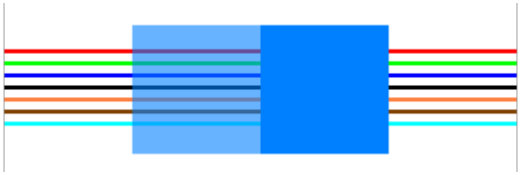


4500ms

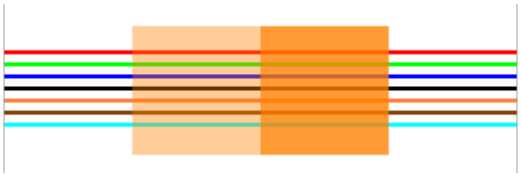
Unexpected Result

If the attribute is ignored or not actioned then the squares differ. For the first two the lines are not visible on the right hand side.

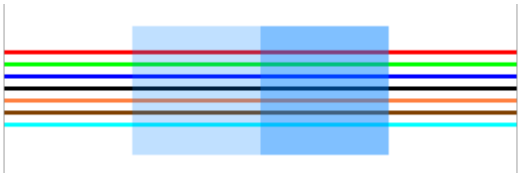
0ms



1500ms



3000ms



4500ms

End of Document